Razvoj algoritma za generiranje beskonačne računalne igre s GPU ubrzanjem

Mustač, Kristijan

SVEUČILIŠTE U ZAGREBU
GRAFIČKI FAKULTET

KRISTIJAN MUSTAČ

RAZVOJ ALGORITMA ZA GENERIRANJE BESKONAČNE RAČUNALNE IGRE S GPU UBRZANJEM

DIPLOMSKI RAD

Zagreb, 2016

Sveučilište u Zagrebu
Grafički fakultet

KRISTIJAN MUSTAČ

RAZVOJ ALGORITMA ZA GENERIRANJE BESKONAČNE RAČUNALNE IGRE S GPU UBRZANJEM

DIPLOMSKI RAD

Mentor:
doc.dr.sc. Tibor Skala

Student:
Kristijan Mustač

Zagreb, 2016

SAŽETAK

Sveprisutna pojava i razvoj raznog softvera pa tako i hardvera doprinijela je globalnom usavršavanju tehnologije na svim područjima i sferama umjetnosti i tehnologije. Digitalni alati za izradu i obradu grafičkog sadržaja jako su usavršeni i napravljeni upravo tako da bi ubrzali procese izrade, u ovom slučaju, animacije. Kako je današnja animacija dosegnula procvat u raznim vizualnim stilovima izrade i načina prikazivanja, tako je i programiranje integrirano u svemu što se koristi u današnje doba. Nedvojbeno postoje razni algoritmi te softveri koji su mozak i pokretačka snaga svake ideje koja je kreirana za pojedinu svrhu i primjenjivost u društvu. Spoj umjetnosti i tehnologije danas je izravno i umjereno sredstvo publiciranja i marketinga u svakoj industriji, koja čak ni ne mora biti usko vezana za vizualno područje prakse rada. Dakako, kakvoća i dosljednost algoritama ovisna je i o kvalitetnom integriranju u sustav kojeg će pokretati, ali također i o samom načinu pisanja koda algoritma. Razvoj beskonačnog algoritma i njegova učinkovita uporaba bit će prikazana tijekom uporabe same računalne igre. U završnoj fazi izrade računalne igre beskonačni algoritam je testiran s različitim brojem ključnih ulaznih parametara (postignuto vrijeme, postignuti rezultat, brzina odvijanja) kako bi se prikazao utjecaj istih.

KLJUČNE RIJEČI: animacija računalne igre, objektno orijentirano programiranje,računalna igra, GPUubrzanje, beskonačan algoritam

ABSTRACT

The ubiquitous occurrence and development of various software including hardware contributed to the global development of technology in all areas and spheres of art and technology. Digital tools for creation and processing of graphic content are highly perfected and made exactly to accelerate and streamline processes of making, in this case, animation. How nowadays animation has reached it's peak status in different visual styles of making and method of display, so did the programming integrated in all that is used in nowadays. Undoubtedly there are various algorithms and software that are the brain and the driving force of every idea that was created for a specific purpose and applicability in society. Blend of art and technology today is an irreplaceable mean of publishing and marketing in each industry, which does not have to be closely related to the visual area of labor practices. Certainly the quality and consistency of an algorithm is dependent on the quality and integration of the system that it runs on but also on the manner of writing the code itself. The development of the infinite algorithm and its efficient use will be displayed during the use of the computer game. In the final stage of the game infinite algorithm will be tested with different number of key input parameters (time achieved, the result achieved, the speed of the same) to show the impact of the same.

KEY WORDS: computer game animation, object oriented programming, GPU acceleration,computer game, endless algorithm

SADRŽAJ

1. UVOD

Informacijske vještine svakog pojedinca današnjice dosegle su velike razine informatičkog razumijevanja i međuljudskog komuniciranja u različitim poljima nauke, pa tako ne samo u znanstvenim nego i društvenim – zabavnim. U radu je prikazan razvoj i upotreba algoritma koji omogućuje tok igre prema ulaznim podacima prethodne razine. Trenutno na tržištu ima dosta aplikacija i programa koji mogu izvoditi razne animacijske radnje, te programa i aplikacija za pokretanje i testiranje kodova. Programi u radu, rabljeni za animaciju i programiranje korišteni su ponajviše zbog pristupačnosti, lakoće izvođenja i funkcionalnosti. Vizualni prikaz i implementacija koda računalne igre potrebne za pokretanje na računalu, iznesena je segmentirano i jasno. Izrada same animacije ove 2D *side-scroller* video igre, opisana je kroz program Spine, dok je programski dio proveden u Flash Builderu. Zanimljivo je to što se ovaj rad može provesti kao općeniti naputak kako uraditi i što bi se sve trebalo napraviti za kreiranje 2D računalne igre, koja niti ne mora biti istog žanra i tematike.

2. VIDEO IGRE

Video igra je elektronička igra koja uključuje ljudsku interakciju s korisničkim sučeljem za generiranje vizualne povratne informacije preko video uređaja kao što je to TV zaslon ili monitor računala. Za video igre se ponekad i smatra da su jedan oblik umjetnosti, ali to je umjerena pretpostavka bez nekog temelja. Elektronički sustavi koji se koriste za igranje video igara su poznati kao platforme, a primjeri za to su osobna računala i igrače konzole. Takve platforme mogu biti većih formata poput velikih *mainframe* računala ili osobnih stolnih računala. Ali isto tako u novije vrijeme mogu biti i manjeg formata poput pametnog telefona ili tableta.[1]

2.1. Računalne igre

PC igre, znane kao i računalne igre su video igre igrane isključivo na osobnom računalu (sa posebnim softverom jedinstveno zvanim *emulator*, moguće je igranje video igara sa drugih platformi na osobnom računalu), a ne na nekoj drugoj igraćoj konzoli poput recimo Playstationa. Također u većini slučajeva računalne igre imaju veći ulazni i izlazni kapacitet te i jači rad procesiranja na računalu. Osobna računala nisu napravljena samo za razonodu i video igre, te tako mogu postojati razlike u prikazivanju iste video igre na različitim hardverima, tj. konfiguracijama. Na njima su omogućene neke značajke za programere poput povećane fleksibilnosti i inovacije, emuliranje, stvaranja modifikacija (eng. *mods*), otvoreni hosting za online igre (u kojoj je jedna osoba igra video igre s ljudima koji su u drugom kućanstvu, isl.) koje dodatno upotpunjuju svijet video igara i softvera na osobnim računalima.

2.2. Platforme i žanrovi

Video igre, kao i većina različitih modulacija medija ima različite oblike karakterističnih opisa, tj. žanrova. Žanrovi video igara se upotrebljavaju za

kategoriziranje igara prema njihovoj igrivosti (eng. *gameplay*), tj. načinu pristupa igri prema igraču. U slučaju žanrova, vizualni aspekt igre i njezina priča ne upotrebljavaju se kao opisni elementi. Žanr video igre je definiran skupom *gameplay* izazova, te se klasificira neovisno o njihovom okruženju ili sadržaju igre, za razliku od drugih umjetnosti poput filma ili knjige. Na primjer, igra „pucačina“ (eng. *shooter*) je još uvijek „pucačina“, bez obzira da li se odvija u svijetu mašte, svemiru, stvarnom svijetu ili nekom drugom vremenu.

2.2.1. Side-scroller 2D video igra

.

„Side-scroller“ video igra ili češće 2D *platformer* je video igra u kojoj se igrivost (eng. *gameplay*) tj. akcija promatra iz pogleda sa strane, a likovi na zaslonu općenito se kreću s lijeve strane zaslona na desni (ili manje često s desne strane zaslona na lijevi) kako bi došli do cilja ili izvršili svoju zadanu misiju. Takve igre koriste računalnu tehnologiju pomicanja zaslona. Uobičajena upotreba *side-scroller* video igre je kao *platformer*. Platformer *side-scroller* akcijska video igra obično uključuje skakanje, penjanje i trčanje likova po nekim sprudovima ili platformama kako bi bili provedeni kroz mnogo različitih razina te naposljetku stigli do cilja. Igre kao što su Super Mario Bros, Prince of Persia (original iz 1989 godine) i Donkey Kong su među najpoznatijim *side-scroller* video igrama te vrste.

Igra izrađena u ovom radu spada pod žanr*side-scroller* igre, a kao začetnika tog žanra bitno je spomenuti Super Mario igru (poznata i kao Super Mario Brothers). To je video igra koju je izdala japanska tvrtka Nintendo 1985. godine. Predstavljala je jednu od prvih video igri u kojoj je lik mogao hodati sa strane, tj. u daljinu širine ekrana. Igrica je ušla i u Guinessovu knjigu rekorda kao najprodavanija video igrica svih vremena (izdanje 1999. godine). U međuvremenu je doživjela brojne inkarnacije, među uspješnijima je bio New Super Mario Bros (2006 god.), a glavni lik Mario je postao gotovo i zaštitni znak Nintenda.[2]

3. GRAFIČKA SUČELJA (GUI)

Grafičko sučelje ili skraćenica GUI (eng. *Graphic User Interface*) je grafičko sučelje za upravljanje programima, te time i sastavni dio programa. Program koji ima grafičko korisničko sučelje koristi prednosti računala (ulazne jedinice; miš, tipkovnica, crtača ploča, itd.). Dobro osmišljeno i oblikovano grafičko korisničko sučelje može odriješiti korisnika od učenja složenih naredbi jezika kojim je određen sustav ili program.

Grafičko korisničko sučelje, kao što je to Microsoft Windows i macOS (Apple) imaju neke bazične komponente poput: pokazivača, pokazivačkog uređaja, zaslona, prozora i izbornika.[3]

Većina grafičkih korisničkih sučelja omogućuje izvršavanje naredbi odabirom neke stavke iz izbornika. Osim njihove vizualne komponente, grafička korisnička sučelja također olakšavaju premještanje podataka iz jedne aplikacije u drugu. Dobro i efikasno grafičko korisničko sučelje uključuje standardne formate za prikaz teksta i grafike. Budući da su formati jako dobro definirani, različiti programi koji se pokreću pod zajedničkim grafičkim korisničkim sučeljem mogu dijeliti podatke.

U video igrama, grafička sučelja drugačije su koncipirana. U većini slučajeva korisnik, tj. igrač igra ulogu glavnoga lika. Tu nije važno da li je igra RPG (eng. *role playing game*), MMORPG (eng. *massively multiplayer online role playing game*), pucačina (eng. *shooter*), strategija, *platformer* isl. Grafičko korisničko sučelje se u takvim okolnostima koristi u svrhu prikazivanja parametara koji su bitni za vrijeme igranja igre. Ti parametri olakšavaju i ubrzavaju snalaženje, a time automatski povećavaju zainteresiranost korisnika za igru.

Slika 1. http://invdr.net/001/wp-content/uploads/2014/03/screenshot-1.jpg, Donkey Kong Country: Tropical Freeze (grafičko korisničko sučelje), 2014

U slučajevima „pucačina“, često se prikazuje koliko municije i koje se oružje trenutno koristi, koliko neprijatelja je ubijeno (eng. *kill count*), isl.

Slika 2. https://gamingreinvented.com/wp-content/uploads/2015/09/800px-WiiU_NewMarioU_3_scrn11_E3.png, New Super Mario Bros. Wii (grafičko korisničko sučelje)

U takvim korisničkim sučeljima bitno je naglasiti da korisnik mora vidjeti koliko ima sredstava (novčići, gemovi i sl.), koliko mu je još života ostalo prije završetka trenutnog nivoa (eng. *level*), kakvu nadogradnju lika koristi i koliko vremena mu je još ostalo do kraja igre. To su bitne stvari koje su uvijek prisutne, pogotovo kod *side-scroller*igara.

3.1. Korisničko iskustvo

Korisničko iskustvo (eng. *User Experience*)je pojam koji opisuje rezultat nekog korisnika u interakciji s nekim objektom ili proizvodom. Takvo iskustvo pokušava maknuti probleme koji se stvaraju u interakciji sa proizvodom i uspostaviti jednostavan i logičan kontakt sa korisnikom. Korisnik mora znati u svakom trenutku lako prepoznati gdje se koji dio bitne informacije nalazi, zato su takve informacije pretežito jako naglašene.

Slika 3. http://nebula.ie/wp-content/uploads/2016/05/Level2_1.png, grafičko korisničko sučelje igre Project G

Slika 3. prikazuje izgled igre Project G i odličan je primjer dobrog korisničkog iskustva kada se govori o igrama. Jasan je prikaz protagonista, tj. glavnog lika i u kakvoj je situaciji. Također, jasno je vidljivo koliko još ima života, naoružanja i bodova. Veoma pregledno sučelje i smještaj elemenata unutar igre, koje omogućava dobro korisničko iskustvo.

3.2. Animacija

Današnja fascinacija računalnom animacijom u punom je jeku. Zahvaljujući nesavršenosti ljudskog oka, odnosno njegovoj tromosti, ako se slike u projekciji izmjenjuju dovoljno brzo vidjet ćemo pokret, tj. animaciju. Taj se efekt doživljava već i sa oko deset slika u sekundi, a sa 24 slike, što je uobičajeni standard za filmove, uvjereni smo da gledamo pokret. Glavni pokretač animiranja zapravo je bio fotograf Eadweard Muybridge, a ne crtač, tj. animator.[4]

Video igra napravljena u ovome diplomskom radu povezana je s digitalnom animacijom. Unutar igre svaki lik ima svoju vrtnju (eng. *loop*) sličica koje čine specifičan pokret, tj. animaciju. Pomoću koda određujemo koliko puta želimo tu vrtnju ponoviti u kojem trenutku i u kojem vremenskom trajanju, te kako je želimo pokrenuti (npr. pomoću interakcije korisnika, nasumično ili nekim događajem unutar igre). Pravi efekt pokreta doživljavamo sa 24 sličice u sekundi, unutar igre neki likovi su animirani sa 15 sličica u sekundi, bez gubitka izgleda fluidnosti.

3.3. Proces animiranja

Tradicionalno animirane produkcije baš kao i drugim oblicima animacije, obično počinju život kao knjiga snimanja. Nacrt napisan sa slikama i riječima, koji podsjeća na ogromni strip. Slike omogućuju animacijskom timu planiranje tijeka radnje i kompoziciju slike. *Storyboard* (plan za animiranje sadržaja) umjetnici će imati redovite sastanke s redateljem, a možda će morati ponovno iscrtati slijed sekvenci nekoliko puta prije nego što konačno dobiju odobrenje.

Prije nego što animacija može početi, preliminarni zvuk (eng. *soundtrack)* se snima kako bi se animacija točno sinkronizirala sa zvukom. S obzirom na spor, metodičan način na koji se tradicionalna animacija proizvodi, gotovo uvijek je lakše uskladiti animaciju na već postojeći *soundtrack*, nego uskladiti glazbu na postojeće animacije. Završni *soundtrack* uključuje glazbu, zvučne efekte i dijalog u

izvedbi glasovnih glumaca. Međutim, pretprodukcijski *soundtrack* obično sadrži samo glasove poput vokalnih pjesama na kojima likovi moraju pjevati zajedno te kao i privremenu partituru. Konačna partitura i zvučni efekti se dodaju u post produkciji.[4]

4. RAZVOJ RAČUNALNE IGRE

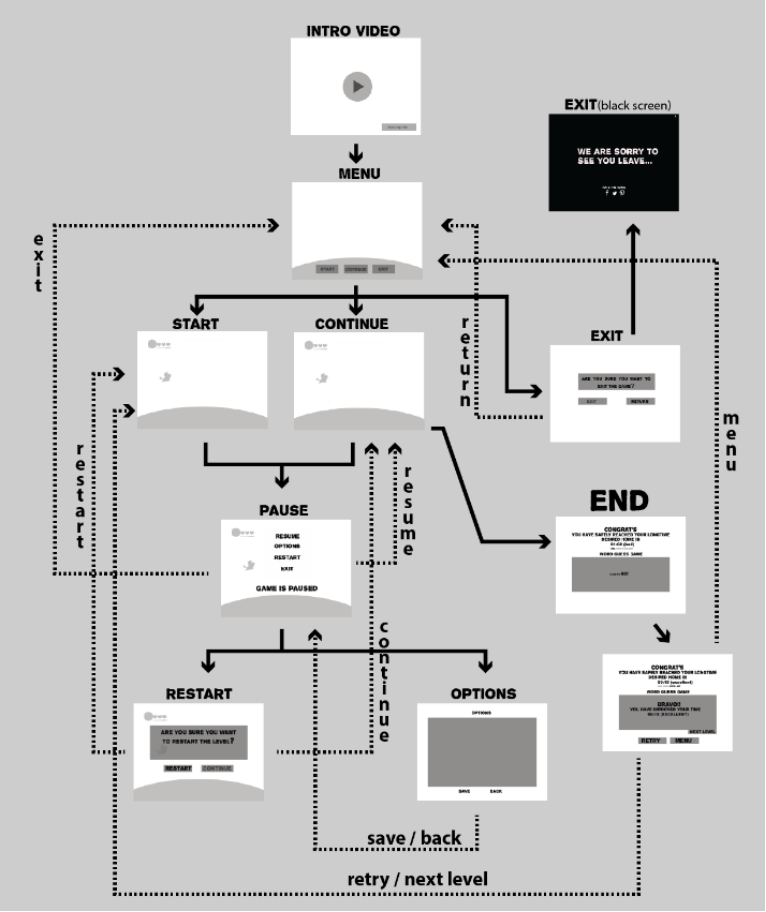
4.1. Opis i ideja računalne igre

Računalna ili video igra koristi jedan ili više ulaznih uređaja. Obično je to kombinacija: tipke i joysticka (igra na arkadi) tipkovnice ili miša (igra na računalu) te kontrolera (igra na konzoli). Iako su video igre preko web preglednika ili kao mobilna aplikacija danas zastupljenije nego prije, zbog jednostavnije izrade i kompatibilnosti, igra je zamišljena kao računalna. Naime, upotreba GPU ubrzanja pridonosi manjem zauzeću radne memorije računala te boljom izvedbom igre.Također, u obzir se može i uzeti mogućnost apliciranja na više platformi (eng. *cross-platform*) tj. operativnih sustava Windows i OS X.

Igrase zove**Strigoforia**, a naziv je došao od porodice sova (glavni lik igre) tzv. noćnih grabljivica (*Strigiformes*). Sova ima 3 života tokom igre, koja se gube pri kontaktu sa neprijateljem, tj. predatorima (jastreb i krtica). Sova je izgubljena i cilj igre je da se sova što prije vrati u sigurnost jazbine, kako ne bi bila plijen navedenim predatorima. Strigoforia je zamišljena kao video igra u borbi sa vremenom. Kako bi što prije stigla u sigurnost jazbine, sova može pojesti skakavca i tako ubrzati svoj let. Konceptualno jednostavnija video igra u koju je uključen algoritam beskonačnog ponavljanja, po završetku razine počinje nova, teža razina koja je definirana završnim parametrima prošle razine.

4.2. Wireframe računalne igre

Bazični vizualni vodič koji prikazuje način funkcioniranja i ponašanja određenih „ekrana“ unutar aplikacije, tj. video igre. Pune linije prikazuju ulaznu radnju korisnika, dok iscrtane linije prikazuju izlaznu, tj. povratnu radnju. To je prva verzija „žičanog“ prikaza (eng. *wireframe*) video igre, koja je prošla nekoliko revizija unutar testiranja grafičkog korisničkog sučelja te korisnikovog iskustva prilikom pokretanja i igranja video igre.

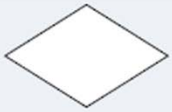


Slika 4. wireframe prikaz računalne igre

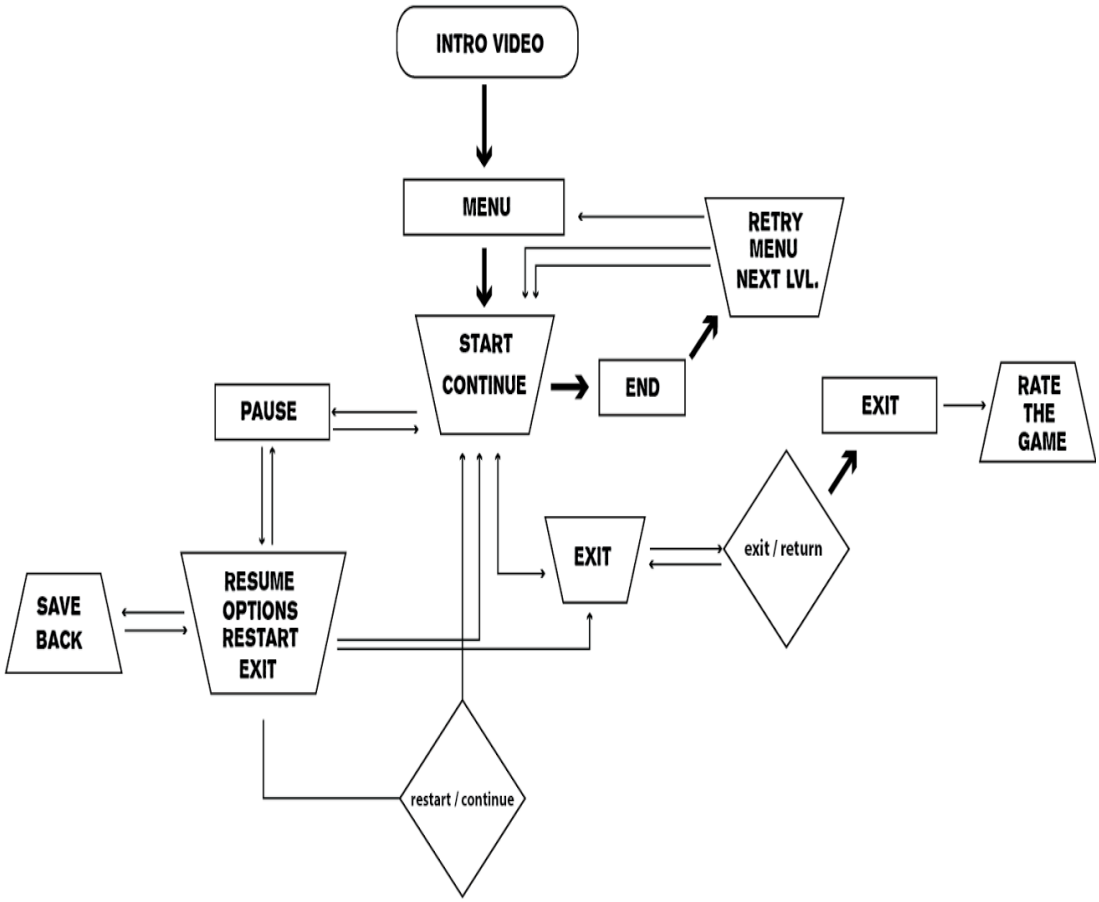
4.3. Dijagram toka

Dijagram toka ili temeljni nacrt je (eng. *blueprint*) vizualni prikaz algoritma (slijed radnji koje dovode do rješenja definiranog problema). Sastoji se od niza simbola povezanih strelicama (vanjskim putevima) koji definiraju tijek i smjer programa.

Osnovni elementi (simboli) koji se rabe su: ulaz, obrada (operacija), petlja (odluka), izlaz i kraj programa.

Simbol	Značenje
	Oval označava početak ili kraj programa. Ako označava početak, u njemu se piše "START", a ako označava kraj programa, piše "STOP"
	Trapez s duljom osnovicom gore označava ulaznu radnju
	Trapez s duljom osnovicom dolje označava izlaznu radnju
	Pravokutnik označava radnju (operaciju) koja je određena programom. To je radnja koja se izvršava jednom naredbom ili odsječkom programa
	Romb označava odluku ili grananje programa
	Strelica označava smjer izvođenja programa između koraka u algoritmu

Slika 5. prikaz osnovnih elemenata toka dijagrama



Slika 6. dijagram toka igre

4.4. Animiranje i priprema materijala za računalnu igru

Strigoforia ima uvodnu scenu, tj. „*intro*“ kao i svaka druga video igra ili njoj srodna umjetnost. Kako glavni lik, ćuk jamar ima neprijatelje poput crveno-repnog jastreba (*Buteo jamaicensis*) tako će u tabli uvoda glavni pokretač bijega sove biti baš on, crveno-repi jastreb. Slika 4. prikazuje crno bijeli prikaz table uvoda, dok će u završnoj obradi biti u boji.

Slika 7. tabla uvoda

Za izgled sove upotrebljavale su se fotografije pravog ćuka jamra iz prirode. Izgled u video igri pokušava biti što realniji stvarnom obliku i izgledu ptice.

Slika 8.
https://upload.wikimedia.org/wikipedia/commons/thumb/5/5a/Brazilian_burrowing_owl_(Athene_cunicularia_grallaria).jpg/220px-Brazilian_burrowing_owl_(Athene_cunicularia_grallaria).jpg, Ćuk jamar

Slika 9. skica glavnog lika

Neke odlike ćuka jamra su naglašenije u video igri (npr. obrve i oči) kako bi ptica došla do većeg izražaja.

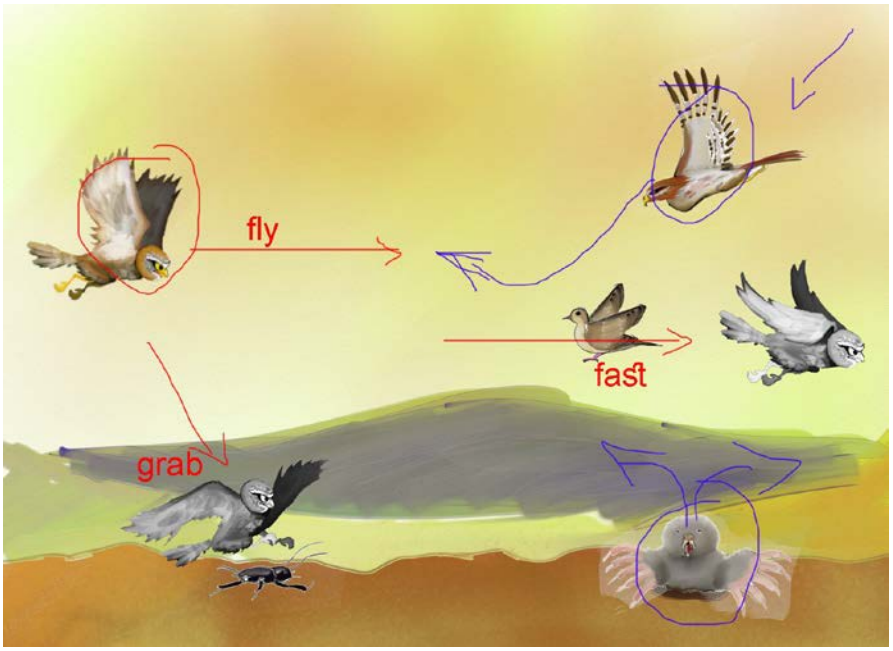
Slika 10. skica glavnog lika

Slika 11.raspodjela grafičkih elemenata unutar videoigre

Na slici 11. prikazana je temeljna raspodjela grafičkih elemenata u video igri. U gornjem lijevom kutu pod brojem 1. nalaze se životi glavnog lika (nakon gubitka trećeg života igra završava), dok se na broju 2. nalazi glavni lik. Brojevi 3,4 i 5 prikazuju glavni izvor hrane (skakavac) i neprijatelja (crveni jastreb i krtica) u video igri.

4.4.1. Animiranje i izrada likova i karaktera računalne igre

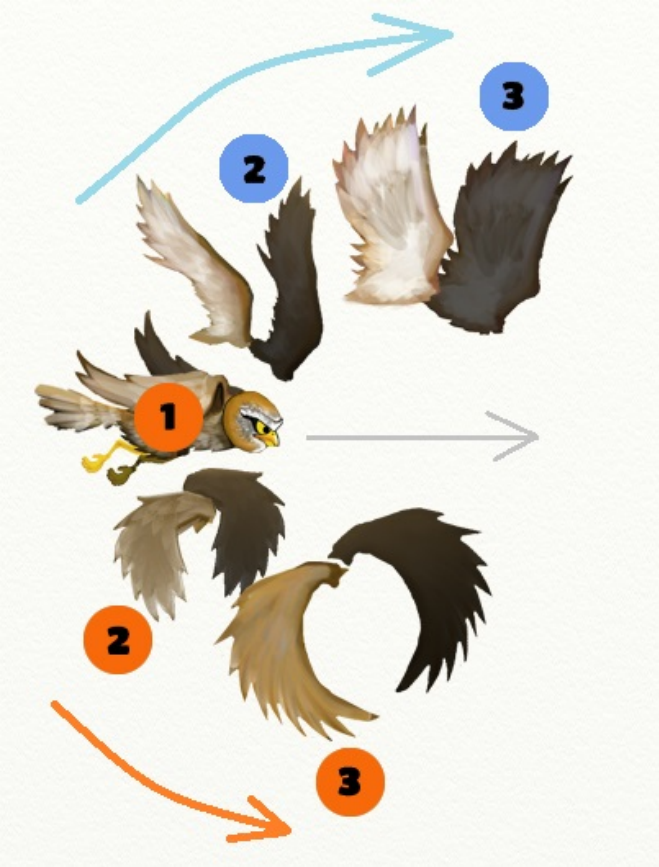
Prvo se trebala postaviti moguća kretnja glavnog lika, sove i neprijatelja kako bi se prema tome odredilo kako će tko reagirati i biti animiran unutar video igre. Na slici 12. vidi se mogući let, tj. kretnja svakog pojedinog elementa unutar video igre, koji je na kraju i usvojen. Crvena boja strelica na slici prikazuje tri načina kretnje (normalno i brzo letenje, te poza u kojoj sova grabi hranu s tla) sove koja su moguća igraču. Plava boja strelica prikazuje kretnju neprijatelja.



Slika 12. mogući načini kretnje

Prikaz koraka animiranja leta sove (*Slika 13*) pokazuje jednostavan pristup problemu animiranja. Glavno je naznačiti na grafičkome elementu lika koji dijelovi se kako trebaju kretati i u kojem smjeru, te u kakvom uzorku (eng. *pattern*) putanje. Također se treba znati kakva kretnja se želi postići, jer nije isto da li lik trči ili skače.

Nakon postavljanja kretnji i kreiranja likova u programima za obradu grafičkih elemenata (Adobe Photoshop i ArtRage 4.5), ti se elementi uvoze u program za animiranje, koji je u ovom slučaju program Spine.

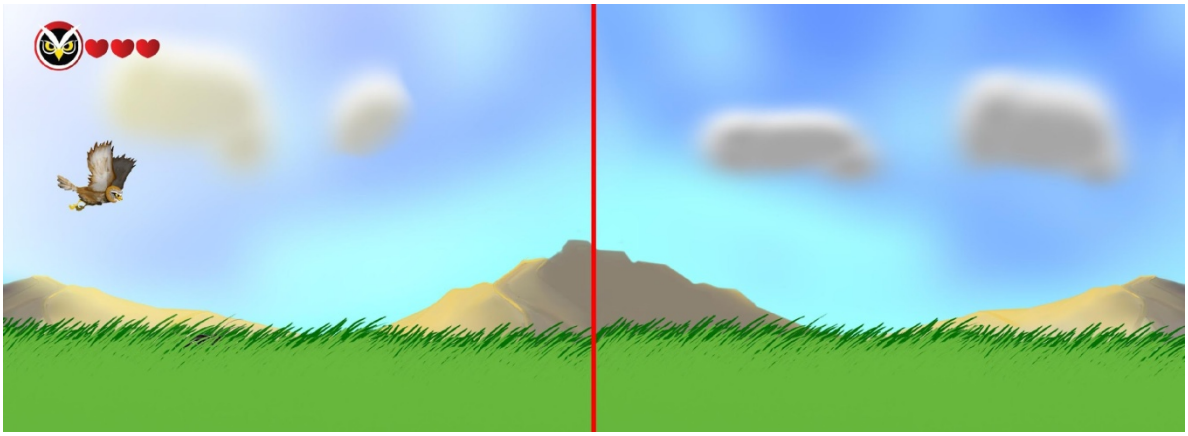


Slika 13. koraci animiranja leta sove

4.4.2. Animiranje i izrada okoline računalne igre (parallax efekt)

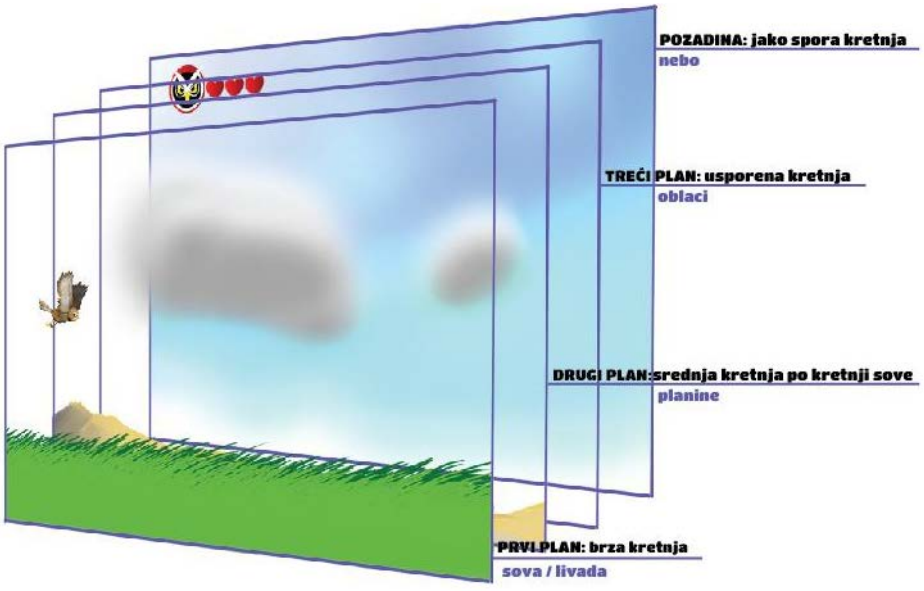
U igri se rabi okolina sa više stadija kretnje. Za to je potreban parallax efekt. Parallax efekt je tehnika pomicanja u računalnoj grafici i web dizajnu, gdje se pozadinske slike pomiču sporije nego slike u prednjem planu, što stvara iluziju dubine u 2D sceni, te omogućuje bolju dubinu efekta kod gledanja.

Za temeljni prikaz igre na monitoru računala koristiti će se rezolucija od 1366x768 piksela. Crvena linija prikazuje do kuda i kako će izgledati igra na prikazu ekrana te rezolucije (*slika 14*).



Slika 14. prikaz parallax efekta na rezoluciji 1366x768 piksela

Sa određenom rezolucijom prikaza igre na ekranu trebala se i odrediti i separacija pojedinih slojeva (eng. *layer*) pozadine kako bi se lakše mogla uvesti u softver za programiranje. Tako je pozadina igre raspodijeljena u 4 sloja. Prvi sloj će sadržavati prikaz livade i on će imati najbržu kretnju unutar igre. Drugi sloj će prikazivati planine i njegova kretnja će biti srednje brzine. Treći sloj će prikazivati oblake i njegova brzina će biti ustanovljena jako sporom kretnjom. Zadnji i najsporiji sloj će prikazivati nebo. Grafički prikaz se vidi na slici 15.



Slika 15. separacija parallax efekta za igru

4.4.3. Animacijski alat (2D Spine)

Spine je animacijski alat koji se fokusira isključivo na 2D animaciju za igrice. Spine ima za cilj učinkovit, dinamičan tijek rada kako za stvaranje animacije tako i za korištenje tih animacija u igricama. Može se isprobati u *demo* verziji jer nije besplatan, tj. *freeware* program.[5]

Animacija u Spineu se stvara dodavanjem slika kostima i zatim animiranjem kostiju. Ovo ima veliku prednost pred klasičnom kadar po kadar animacijom i to ne samo zbog brzine izrade animacije već i sljedećih stvari:

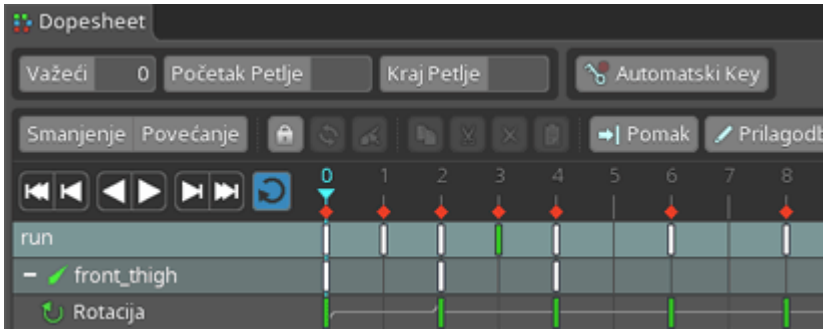
Manja veličina jer klasična animacija zahtjeva sliku za svaki kadar animacije. Spine animacija pohranjuje samo podatke kostiju, koji su mali, te omogućava pakiranje igrice bogate jedinstvenom animacijom.

Umjetnički zahtjevi koje Spine animacija zahtjeva lakše i brže su implementirani, te manjih dimenzija jer pohranjuju zapis koji ih opisuje efikasno upotrebljavajući vrijeme korisnika.

Spine animacija također koristi interpolaciju, tako da je animacija uglađena kao i kadar po kadar. Slike vezane za kosti mogu biti zamijenjene kako bi odgovarale likovima s različitim stvarima i efektima. Ta ista animacija može biti ponovno upotrijebljena za likove koji izgledaju različito od prvobitnog (efikasno kod više različitih kostima istog lika). Također, animacije mogu biti zajedno stopljene. Na primjer, likovi mogu izvoditi animaciju pucanja dok istovremeno izvode animacije hodanja, trčanja ili plivanja.

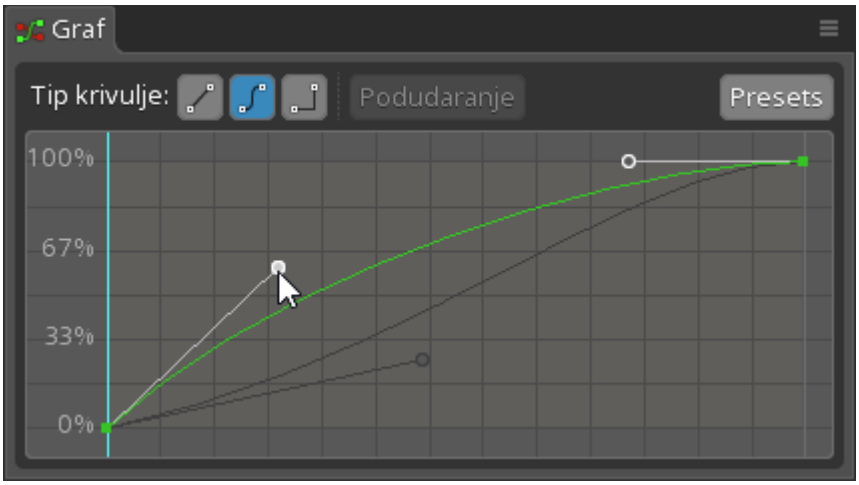
Upravljanje kostima može se ostvariti kroz kod koji omogućava efekte poput pucanja prema poziciji kursora miša, gledanje prema neprijateljima u blizini ili naginjanja prema naprijed pri trčanju uzbrdo.

Spine ima i neka specifična svojstva vezana uz lakšu izradu animacije. Vremenski alat (eng. *Dopesheet*)omogućuje detaljan pregled cijelog vremenskog toka (eng. *timeline*) koji stvara animaciju i omogućava da se naprave precizne prilagodbe u animacijskom vremenskom tijeku.



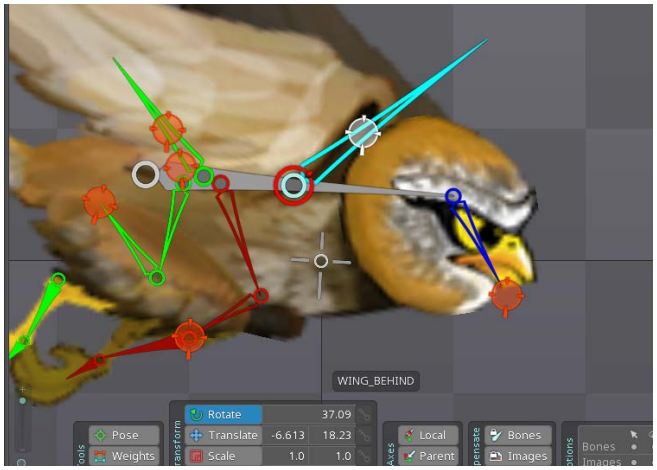
Slika 16. prikaz funkcija vremenskog alata unutar Spine-a

Grafički urednik definira bezierovu krivulju za interpolacije između zadanih točaka (eng. *key*), pružajući fluidniju i oku ugodniju kretnju.



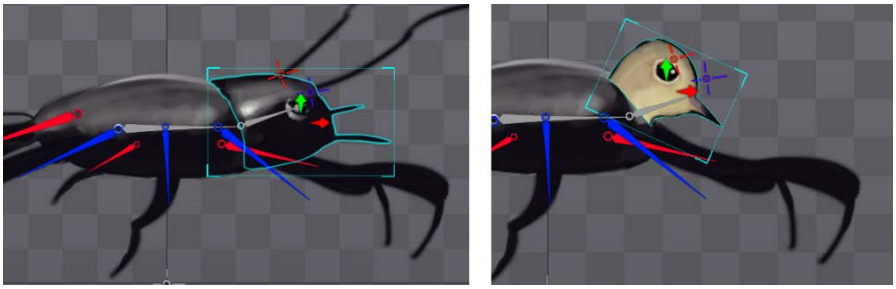
Slika 17. prikaz grafa krivulje

Alat obrnute kinetike (eng. *inverse kinematics*) poze koristi inverznu kinematiku za brzu postavu kostura.



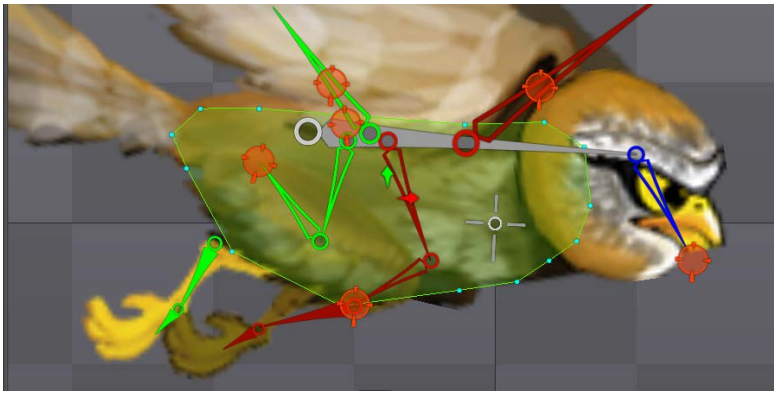
Slika 18. prikaz upotrebe obrnute kinetike

Opcija mijenjanja kostima (eng. *skinning*) omogućava individualnim vrhuncima u mreži da budu spojeni na različite kosti, te time i prebacivanje između setova animacije. Oni omogućavaju organizaciju za dodatke i omogućavaju animacije koje mijenjaju dodatke da se ponovno koriste pri različitim likovima. Kada se kost pokreće, vrhunci se kreću s njima i mreža se automatski deformira. Postavljanje lika sa slikama koje se mogu savinuti postaje jednostavno kao i postavljanje kostiju.



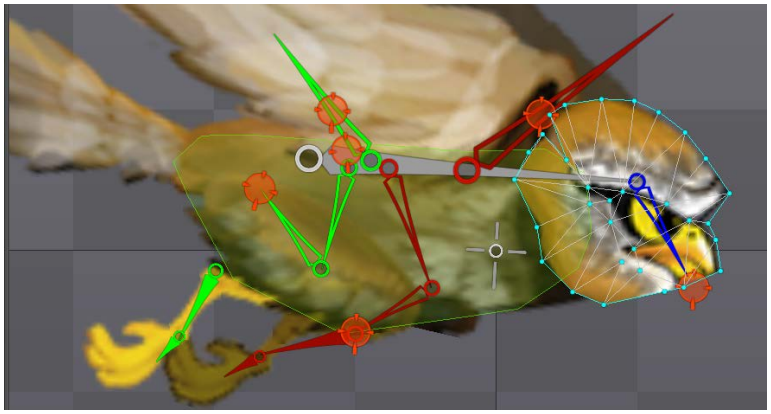
Slika 19. prikaz funkcije „skinninga“

Funkcija geometrijskog ocrtavanja (eng. *bounding box*) je opisana poligonom koji je spojen na kost. Poput slika, poligon se pokreće kako se kreće i kost. Može se koristiti za ugađanje detekcije i integracije fizike.



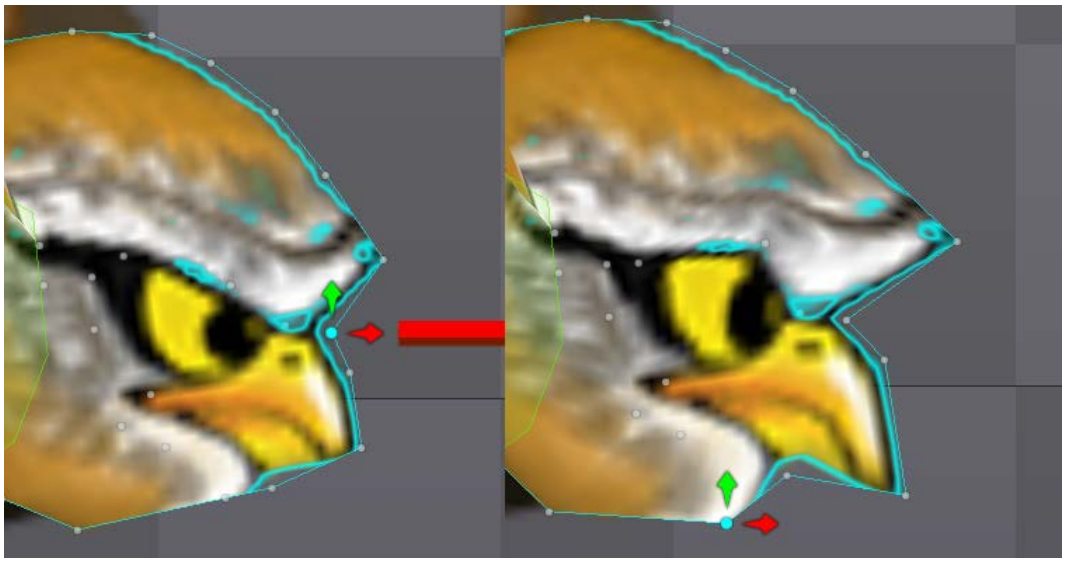
Slika 20. prikaz „bounding box-a“

Također umjesto iscrtavanja pravokutnika, mreže omogućavaju da se specificira poligon unutar vaše slike. Ovo poboljšava ispunu, jer pikseli izvan poligona neće biti iscrtani, što je posebno važno za igre na mobitelima. Mreže također omogućavaju FFD (eng. *free form deformation*) i „*skinning*“.



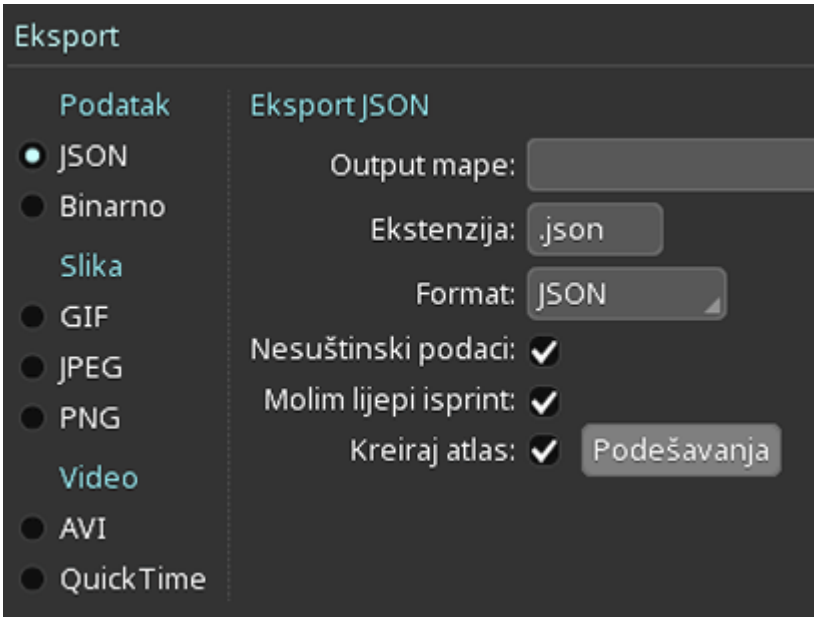
Slika 21. prikaz izgleda deformacije bezgranične forme

Deformacije bezgranične forme (FFD) dozvoljavaju kretanje individualnih mrežnih vrhunaca da deformiraju sliku. FFD dozvoljava mreži da se rastegne, spljošti, svije i poskakuje na način na koji pravokutne slike inače ne mogu.



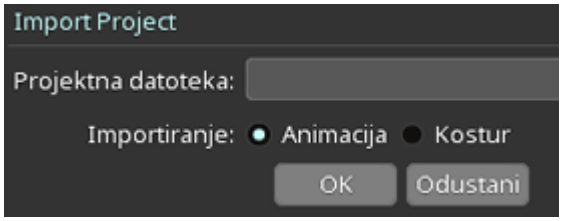
Slika 22. prikaz upotrebe „FFD-a“

Spine izvozi (eng. *exporting*) animacijske podatke u svoj dokumentirani JSON i binarni format što je idealno za korištenje u Spine *runtime* bibliotekama. Spine također može eksportirati animirani GIF, PNG ili JPG slijed slika i AVI ili QuickTime video.



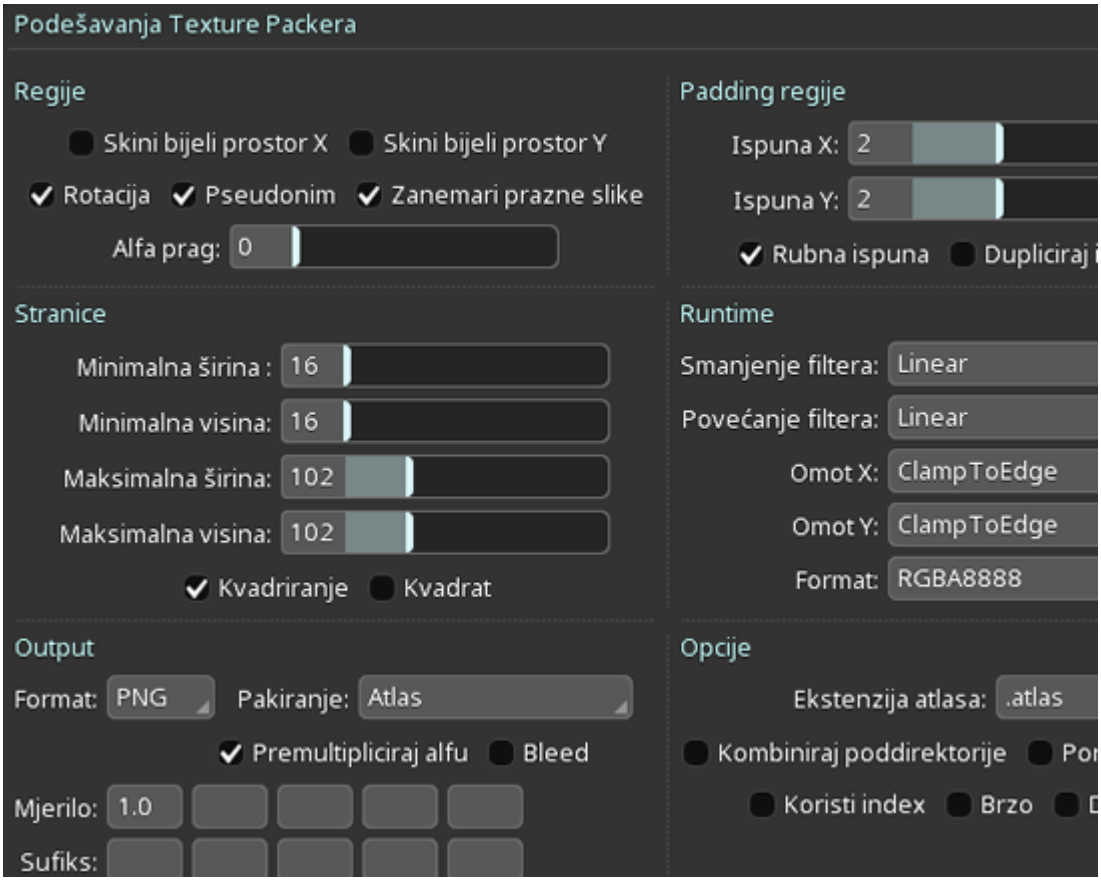
Slika 23. prikaz prozora za izvoz dokumenata unutar Spine-a

Spine može uvoziti (eng. *importing*) podatke u svom JSON ili binarnom formatu, koji omogućava stazu za uvoz podataka iz drugih alata u Spine. Kosturi i animacije se također mogu uvoziti iz drugih projektnih dokumenata.



Slika 24. prikaz prozora za uvoz dokumenata unutar Spine-a

Spine može pakirati slike u teksturne atlase ili *spritesheetove* koji rezultiraju efikasnijim iscrtavanjem (eng.*rendering*) igre. Spine *texture packer* ima različite odlike kao što su uklanjanje bijelih slika, rotacija, automatsko mjerilo i drugo.



Slika 25. prikaz prozora „texture packer-a“

Dizajniranje sjajnih animacija je samo dio zadatka izrade igre. Te animacije napravljene unutar Spine-a trebaju se također renderirati unutar igre. Spine pokretačke biblioteke (eng. *runtime library*) omogućavaju da se te animacije učitaju i renderiraju unutar igara, baš kao što to čine i u Spine-u.

Spine pokretačke biblioteke imaju API (eng. *application program interface* – setrutina, protokola i alata za izgradnju softverskih aplikacija koji se najčešće upotrebljava kod interakcije softverskih komponenti grafičkog korisničkog sučelja) koji pruža direktni pristup kostima, dodacima, *skinovima* i drugim animacijskim podacima. Kostima se može proceduralno manipulirati, animacije se mogu miješati, križno nestajati i drugo. Službene pokretačke biblioteke su na GitHubu i

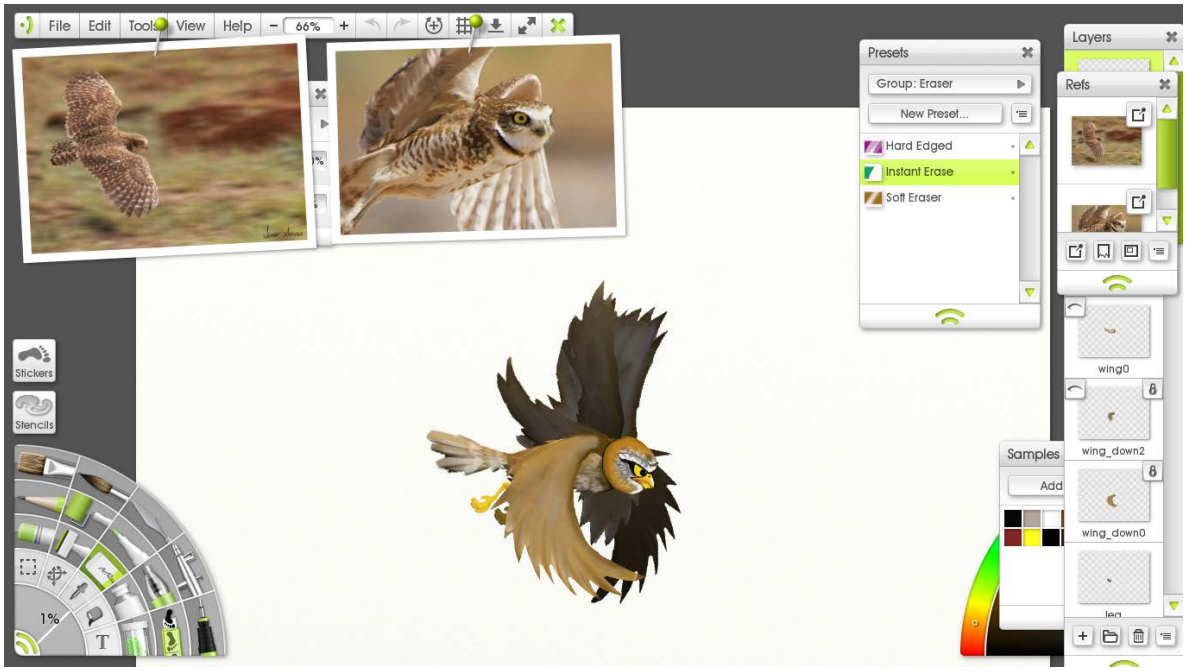
licenciranjem Spine dozvoljava korištenje tih pokretačkih biblioteka unutar igre ili aplikacije. Spine također podržava Starling okvir (eng. *framework*) koji se upotrebljava za izradu igre u ovom radu.

Spine se također aktivno izgrađuje i redovno izbacuje nadogradnje, ponajviše zbog čestih povratnih informacija korisnika.

4.4.4. Softver za izradu i uređivanje likova i okoline (ArtRage 4.5 i Photoshop)

ArtRage je grafički alat (software) koji je namijenjen početnicima i profesionalcima u svijetu digitalnog crtanja i slikanja.[6] Nažalost nije besplatan, ali program ima određeni novitet i raritet na današnjem tržištu (poput programa *Black Ink*). Također ima i veliku bazu korisnika. Ovaj alat imitira tradicionalne slikarske tehnike i pokušava ih prevesti na digitalno platno. U ovom diplomskom radu za izradu potrebnim elemenata u scenama koristio se ArtRage i u nekim poljima je bio daleko iza Photoshopa, dok je u nekima bio daleko ispred.

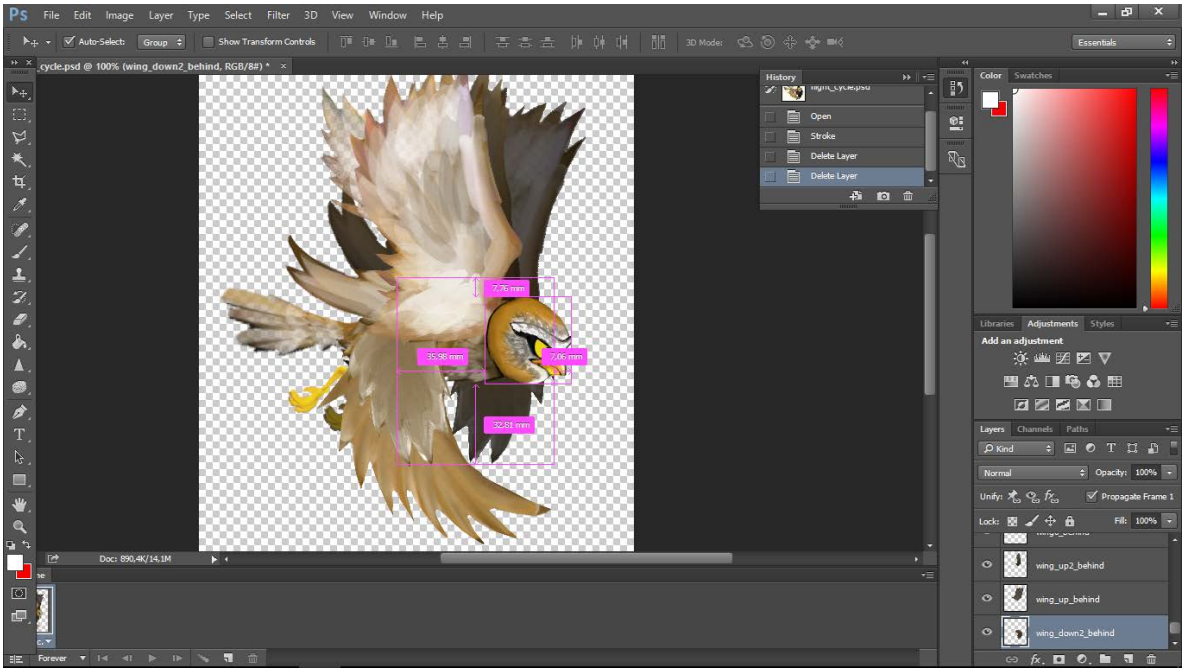
Slika 26. logo programa ArtRage



Slika 27. Sučelje programa ArtRage i jedna od izrađenih motiva za animiranje

Adobe Photoshop je rasterski grafički alat i softver koji je namijenjen početnicima i profesionalcima u svijetu digitalnog crtanja i slikanja, te obrade istih. Napravljen je i publiciran od Adobe Systems za Windows i OS X operacijske sustave. Ovaj program je već desetljećima u samom vrhu grafičke industrije, što i nije nimalo čudno uzevši u obzir koje snažne i brze alate sadrži. Jedno od karakteristika Photoshopa je poznato i jednostavno grafičko korisničko sučelje koje uz sve nadogradnje svake godine zadržava svoj prepoznatljiv i snalažljiv oblik.

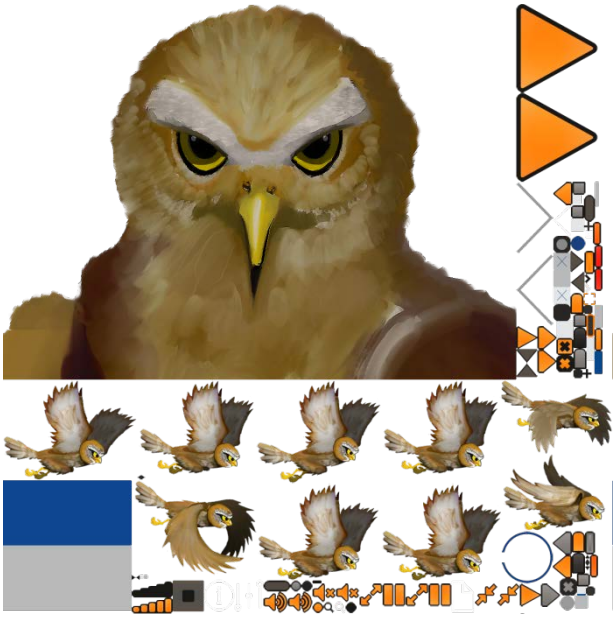
Slika 28. logo programa Adobe Photoshop



Slika 29. Sučelje programa Adobe Photoshopa

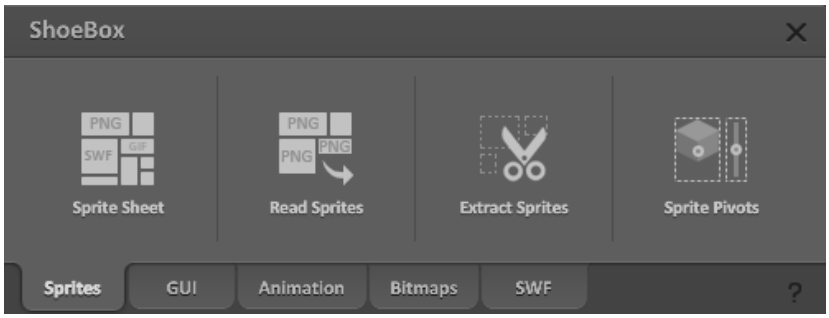
4.5. Priprema materijala za učitavanje u računalnu igru

Za učitavanje vizuala igre morao se napraviti teksturni atlas jednakih omjera (2048x2048 piksela), kako bi igra lakše bila pokrenuta i bolje strukturirana. Slika 8. prikazuje prvu verziju teksturnog atlasa koji se koristi u igri. Unutar takvog atlasa nalazi se cijeli *art* igre, a koji se obično sastoji od: simbola, ikona i raznih grafičkih elemenata. Prema situaciji i interakciji korisnika, tzv. igrača, ti elementi će se neprestano sortirati i birati unutar igre. Time je uvelike olakšano i smanjeno korištenje memorije uređaja na kojemu se video igra pokreće.



Slika 30. prva verzija teksturnog atlasa (eng. spritesheet)

Samo pakiranje ili stvaranje teksturnog atlasa najbolje se i najbrže izrađuje pomoću softvera Texture Packer ili ShoeBox (naravno postoje i drugi softveri). Za ovaj rad se upotrebljavao softver ShoeBox.[7] Samo sučelje softvera je veoma jednostavno za shvatiti, a svaka radnja obavlja se povlačenjem željenih datoteka na specificirani dio sučelja. Unutar sučelja mogu se stvarati teksturni atlasi od različitog broja grafičkih elemenata, također se i sam atlas kao cjelina može „izrezati“ na manje dijelove koji će se otvarati pojedinačno, a ne kao takva cjelina.



Slika 31. sučelje ShoeBoxa

5. RAZVOJNO OKRUŽENJE

5.1. Programsko okruženje

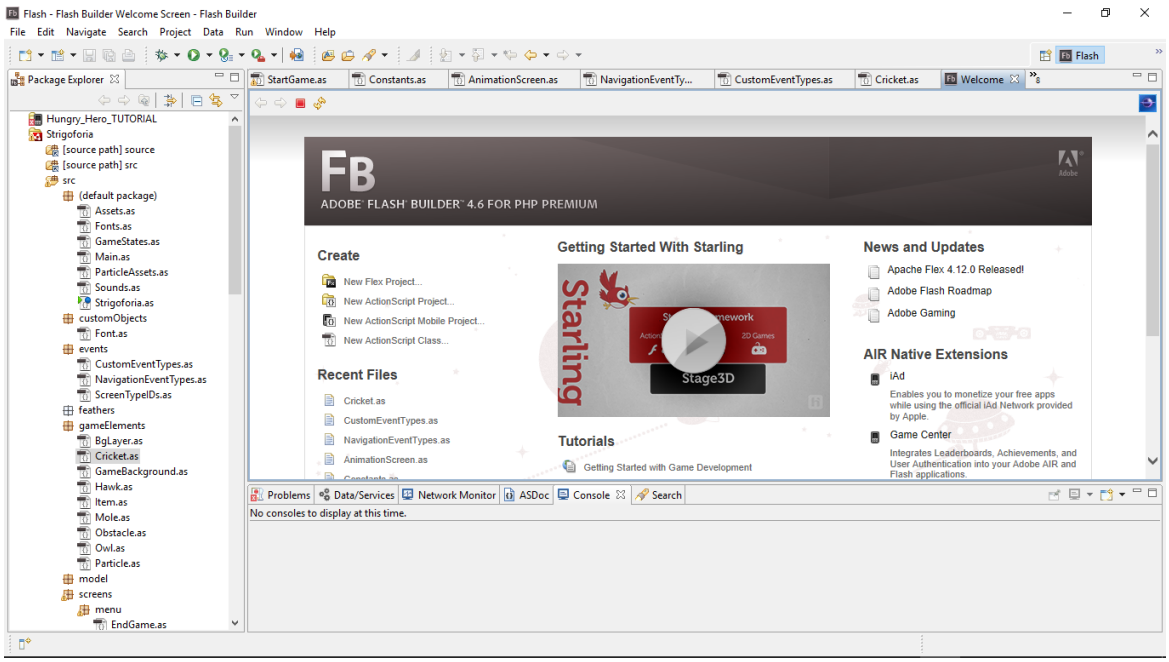
Adobe Flash Builder je integrirano razvojno okruženje (eng. *integrated development environment*- IDE)za izradu *cross-platformi*, bogatih aplikacija za Internet (eng. *rich Internet applications* - RIA), te za stolna računala kao i za veći raspon mobilnih uređaja.[8] Flash Builder u svom razvojnom okruženju ima opcije testiranja, razotkrivanja bugova (eng. *debugging*) te također i alata za profiliranje (eng. *profiling)* koji pomažu za bolju i učinkovitiju razinu efektivnosti i kvalitete rada.

Flash Builder je napravljen, tj baziran na Eclipsu (integrirano razvojno okruženje otvorenog koda, eng. *open-source*) te sadrži sve alate potrebne za razvitak aplikacija koje koriste otvoreni kod poput Flex okvira (eng. *framework*) i ActionScripta 3.0 jezika.

Flash Builder ima potpunu podršku za kreiranje i izradu aplikacija pomoću Apache Flex alata za izradu softvera (eng. *Software Development Kit*). Pri izradi Flex projekta u Flash Builderu može se specificirati upotreba Apache Flex SDK-a. Može se pokrenuti na Microsoft Windowsima i Apple Mac OS X-u, te je dostupan u više verzija. Opcije za konfiguraciju instalacije dopuštaju instaliranje Flash Buildera kao set dodataka (eng. *plug-in*) u postojećem instaliranom Eclipse alatnom sustavu (eng. *workbench)* ili kao zasebnu instalaciju koja uključuje Eclipse alatni sustav.

U Flash Builderu mogu se kreirati aplikacije pomoću Flex okvira (eng. *framework*), MXML-a, Adobe Flash Playera, Adobe AIR, ActionScript 3.0 i LiveCycle Data Services. Pod to spada: izrada Flex projekata koji mogu raditi sa bilo kojom *backend* tehnologijom poslužitelja, uključujući Adobe ColdFusion, LiveCycle Data Services i PHP. Stvaranje ActionScript projekata koji koriste Flash API (ne Flex *framework*). Stvaranje Flex mobilne aplikacije za platformu Google Android i ActionScript mobilnih aplikacija za Apple iOS platforme. Stvaranje prilagođenog koda biblioteke (eng. *library*) koju možete implementirati kao komponentu datoteku biblioteke (SWC – *Shockwave Component*) koja se može

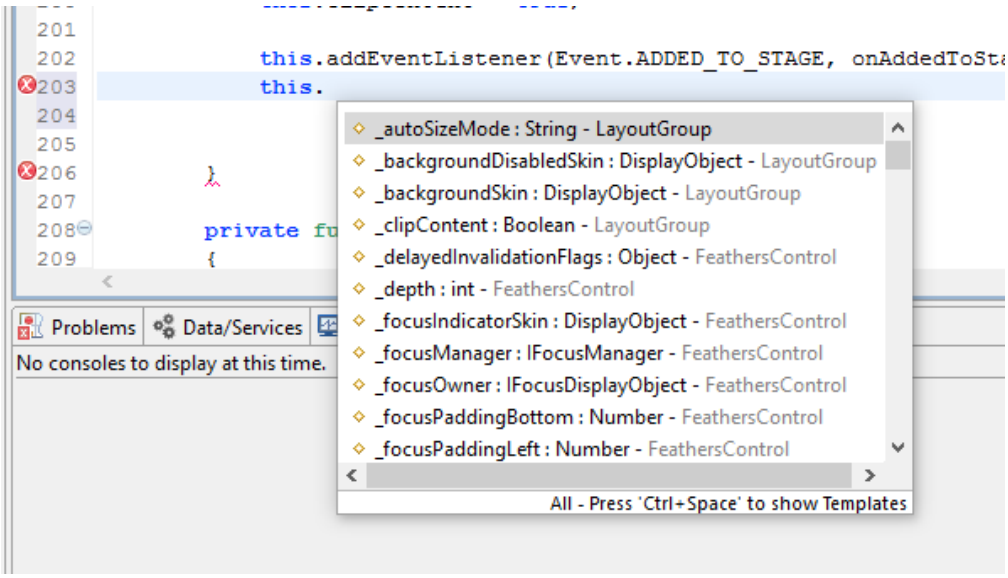
podijeliti među aplikacijama ili distribuirati prema drugim programerima.Stvaranje Flash Professional projekata za uređivanje i ispravljanje Flash FLA (*Editable Adobe Flash*) ili XFL (*Flash Movie Archive*) datoteka stvorenih sa Flash Professional CS5.



Slika 32. prikaz sučelja Flash Buildera

Flash Builder ima potrebne alate za razvoj aplikacija koje kao što je već napomenuto prije, koriste Flex okvir (eng. *framework*) i ActionScript 3.0 jezik. Unutar samog razvojnog okruženja Flash Buildera postoje neka karakteristična svojstva poput: uređivanja i publiciranja koda, upravljanja i izgradnja projekata isl.

U Flash Builderu se mogu pisati i uređivati izvorni kodovi aplikacija pomoću brojnih razvojnih alata. Alati za razvoj koda uključuju restrukturiranje koda (eng. *code refactoring*), aludiranje unutar koda (eng. *code hint*), pojednostavljenu navigaciju koda, automatsku provjeru pogrešaka unutar koda i druge značajke.

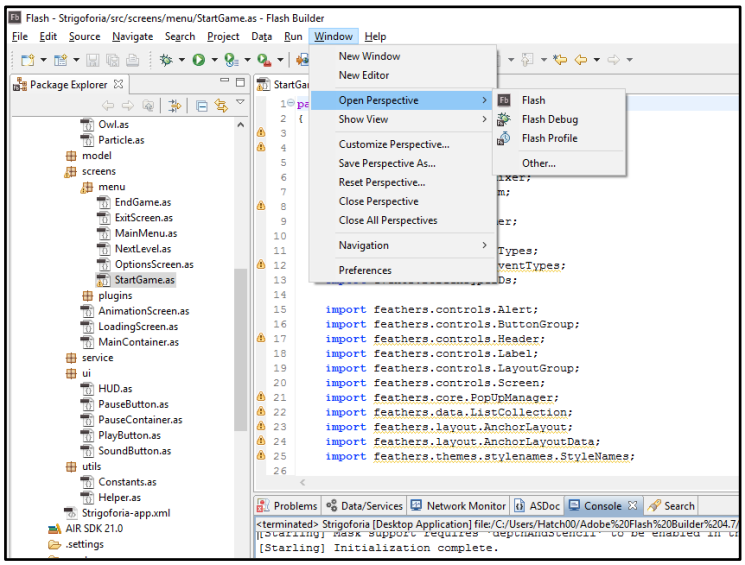


Slika 33. aludiranje unutar koda (eng. code hint)

Publiciranje, tj. izdavanje koda aplikacije kako bi ga korisnici i drugi programeri mogli vidjeti je također veoma lako sa dostupnim alatima.

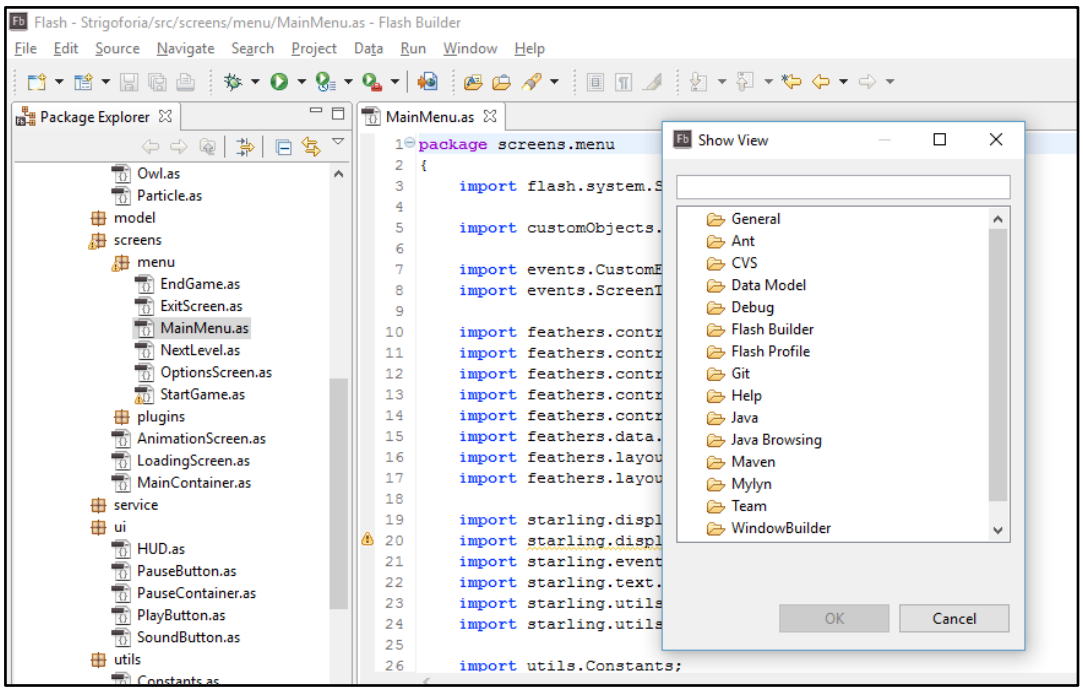
Upravljanje projektima, mapama, datotekama i drugim resursima poput kreiranja, modificiranja i brisanja istih, te povezivanje raznih „resursa“ izvan projekta i tome slično je brzo moguće uraditi, također. Prilagođavanje razvojnog okruženja (eng. *workbench*) Flash Buildera tako da odgovara specifičnim potrebama korisnika.

Pojam „workbench“ se odnosi na razvojno okruženje Flash Builder koje ima sve potrebne alate za izradu aplikacije. To razvojno okruženje ima tri glavna elementa: perspektive (eng. *perspectives*), urednike (eng. *editors*) te poglede (eng. *views*) . Sva tri elementa se simultano upotrebljavaju u različitim procesima izrade aplikacije. Perspektiva je grupa pogleda i urednika unutar razvojnog okruženja. Flash builder sadrži dvije perspektive: Razvojnu perspektivu za razvoj aplikacija (Flash) i perspektivu za razotkrivanje *bugova* unutar aplikacije (Flash Debug).



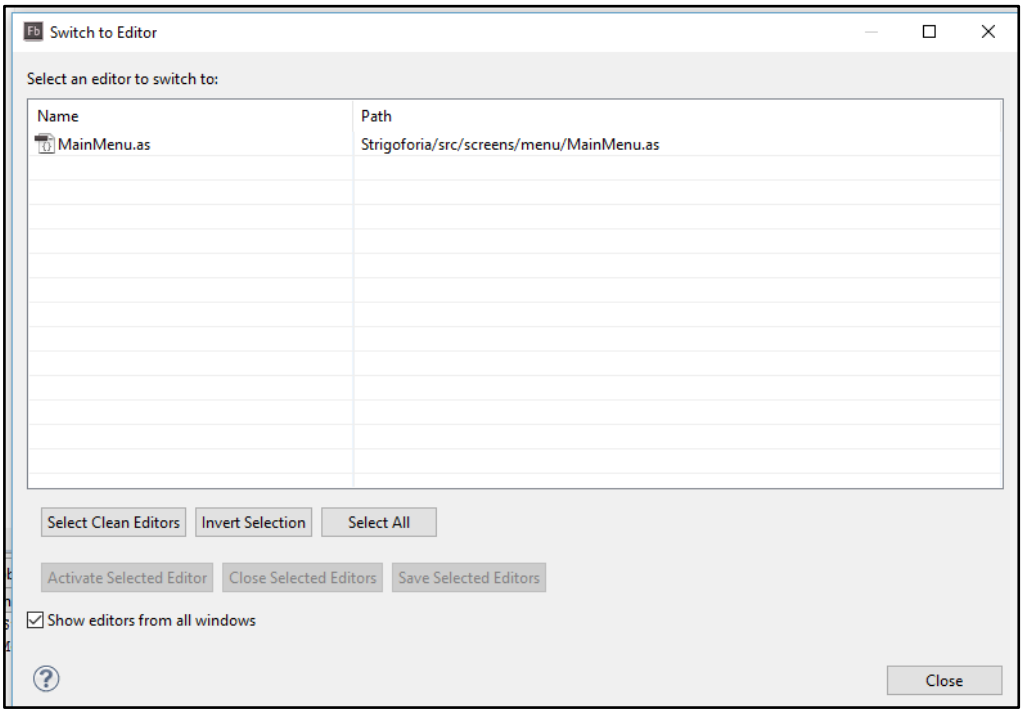
Slika 34. prikaz perspektiva

Ako se Flash Builder koristi u konfiguraciji kao set dodataka, razvojno okruženje može imati dodatne perspektive poput Java perspektive koja ima urednike i poglede koji se rabe za izradu Java aplikacija.[9]



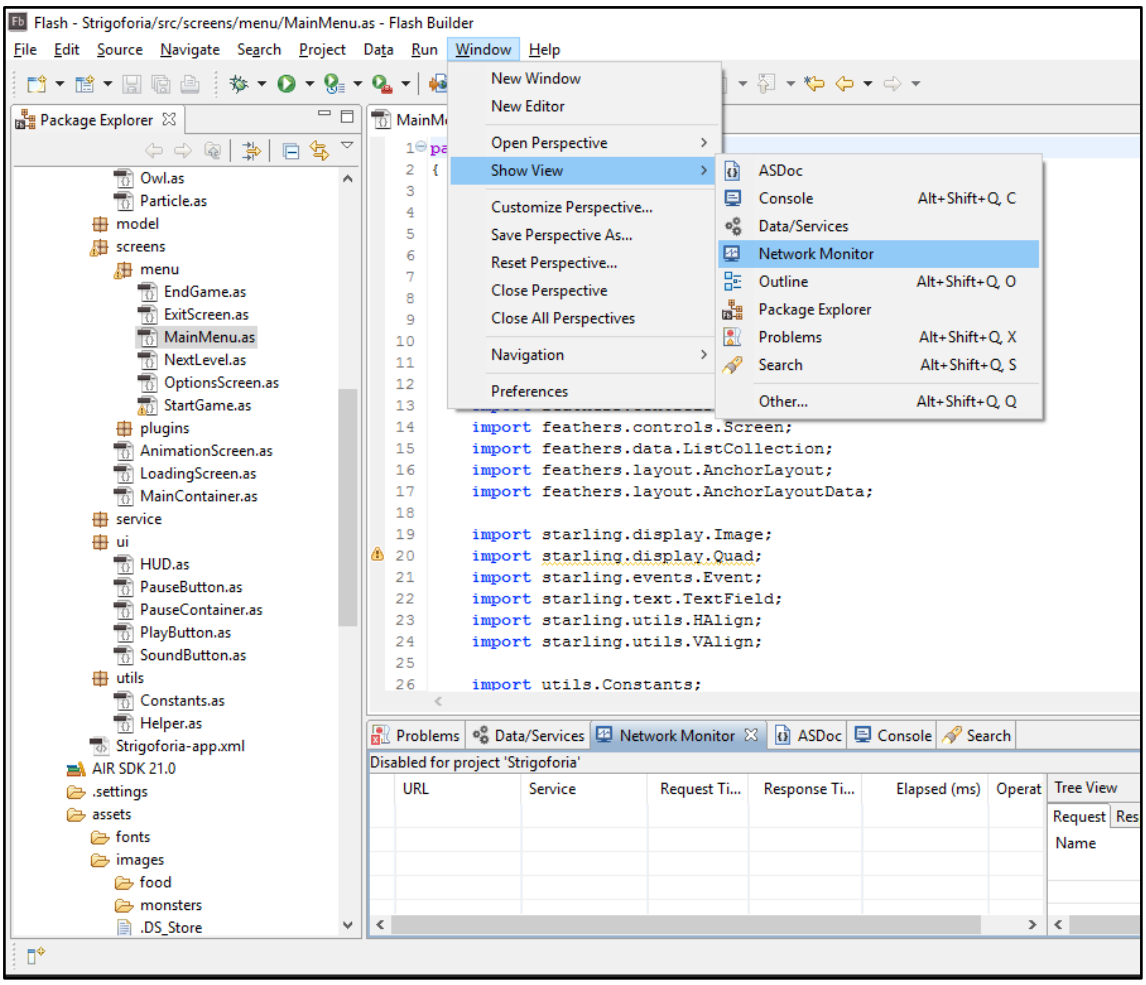
Slika 35. prikaz opcija odabira Java perspektive

Urednik dopušta uređivanje različitih tipova datoteka. Dostupni urednici unutar programa ovise o broju instaliranih Eclipse dodataka. Flash Builder ima urednike za pisanje MXML-a, ActionScript 3.0, i CSS (Cascading Style Sheets) koda.



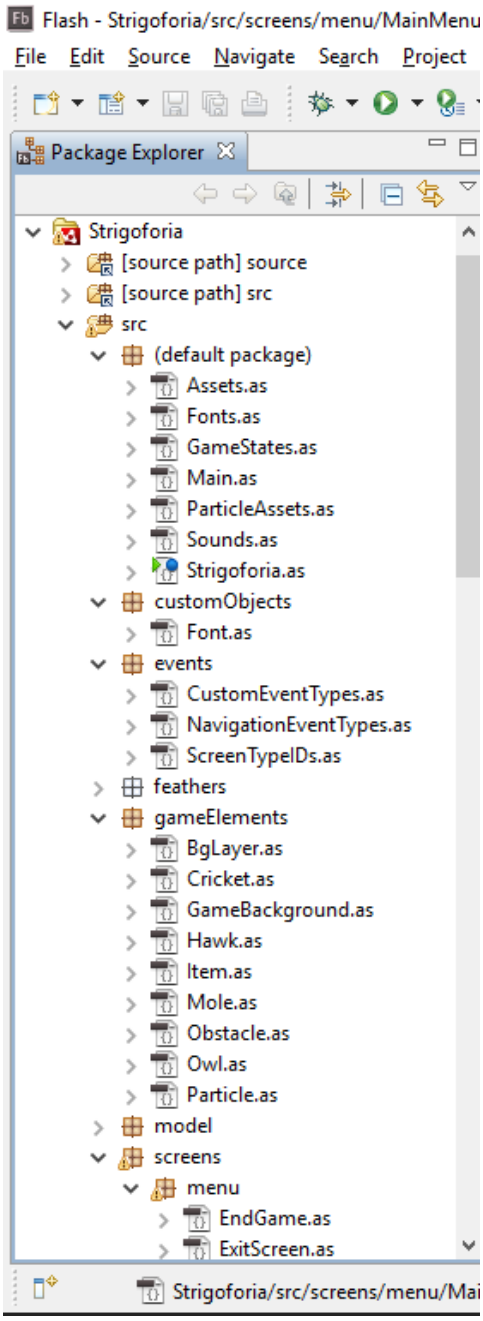
Slika 36. prikaz urednika

Pogled tipično proširuje urednika. Na primjer pri uređivanju ActionScript ili MXML datoteke biti će prikazani podržani pogledi.



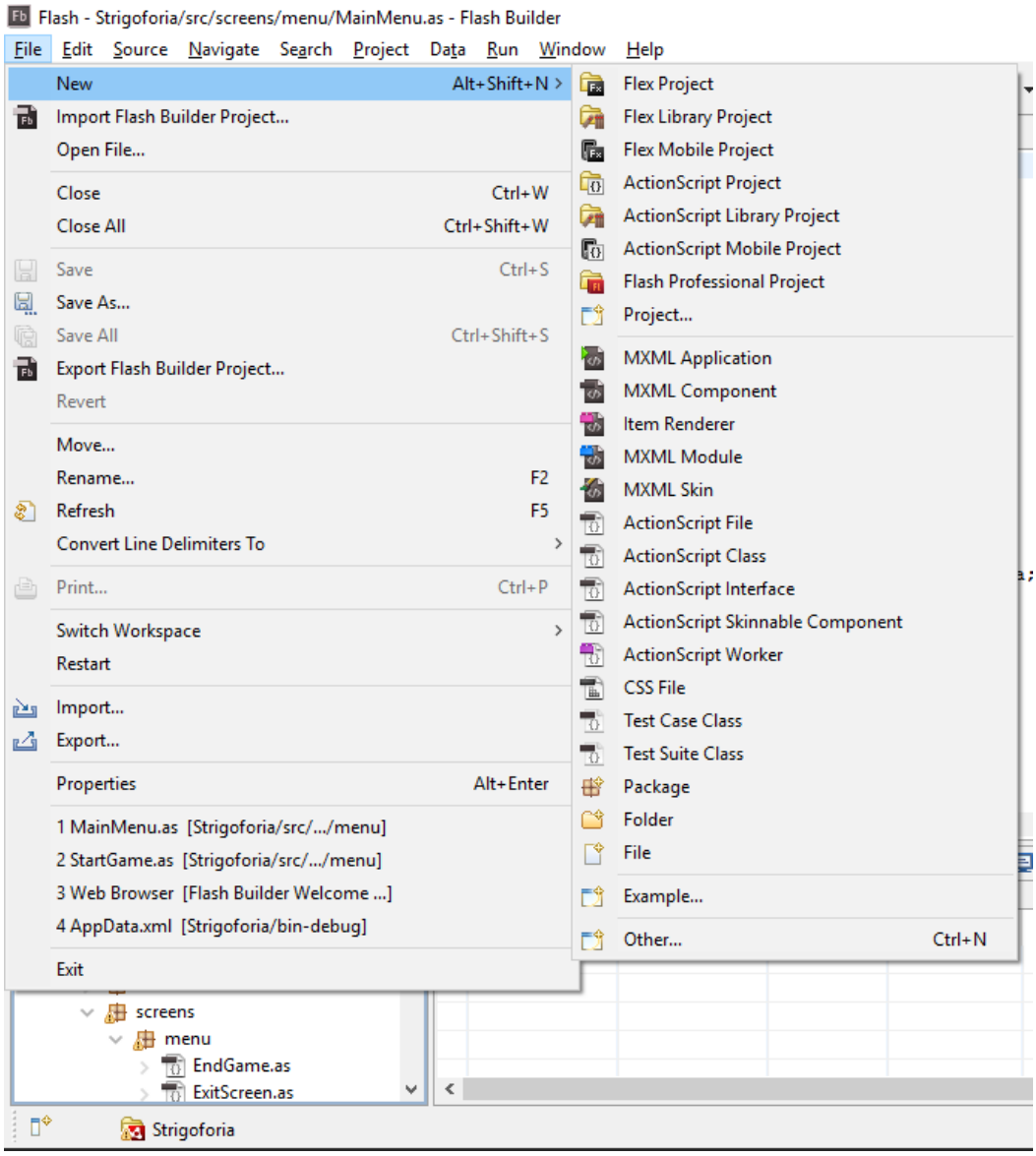
Slika 37. prikaz pogleda

Radno okruženje (eng. *workspace*) je područje U Flash Builderu definirano kao sistemska datoteka koja sadrži resurse (datoteke i mape) koji čine aplikaciju, tj. projekt. U danom trenutku rabi se samo jedno okruženje, ali je moguće označavati više radnih okruženja svaki put kada se Flash Builder upali.



Slika 38. prikaz radnog okruženja

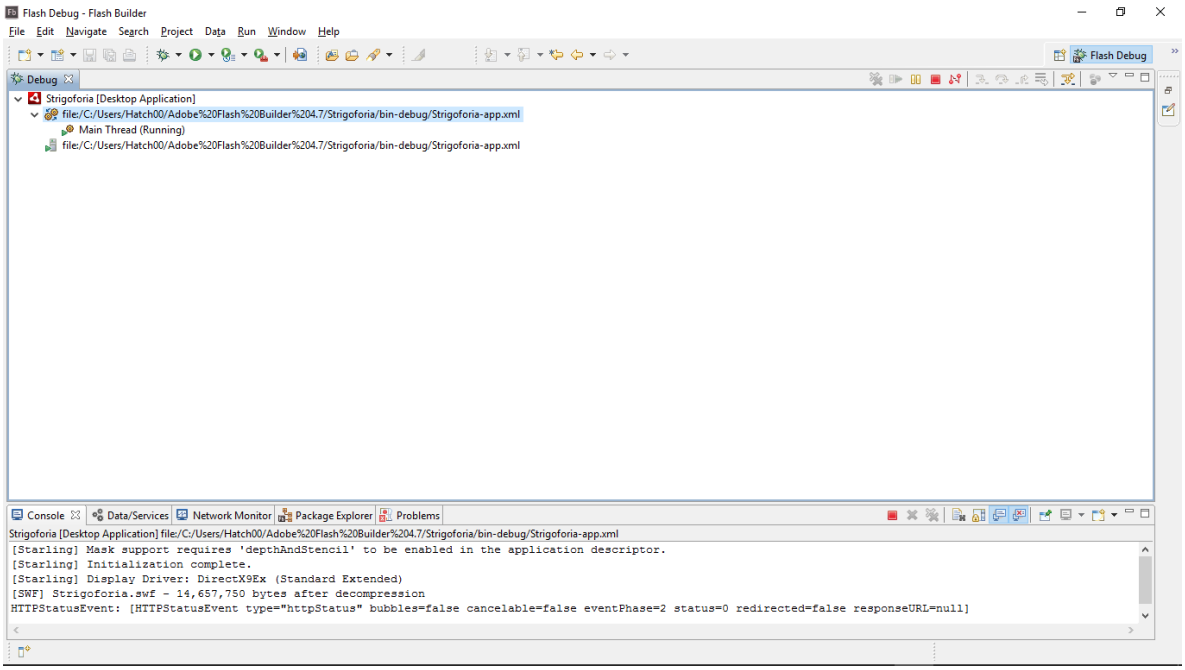
Svi ti resursi (datoteke i mape) koji čine aplikaciju nalaze se unutar projekta. Nije moguće napraviti aplikaciju bez da se prvo kreira projekt. Flash Builder podržava različite tipove projekata, sve ovisno o aplikaciji koja se radi.



Slika 39. prikaz mogućih resursa

Flash Builder automatski sažima (eng. *compile*) i gradi aplikaciju za razotkrivanje *bugova* ili za produkcijsko izdanje. Razotkrivanje *bugova* unutar aplikacije je inače zamoran posao, ali sa integriranim alatom za razotkrivanje *bugova* to je ležeran i informativan posao, a aplikacija može biti pokrenuta

tetestirana na stolnom računalu ili mobilnom uređaju. Također Flash Builder može pokrenuti aplikaciju u web pregledniku, Adobe AIR-u ili u običnom Flash playeru. Mogu se napraviti proizvoljne konfiguracije za pokretanje kako bi se stvorili uvjeti za kontrolirano pokretanje aplikacije.



Slika 40. razotkrivanje bug-a

Identificiranje točaka najvećih gubitaka memorije i izvođenja aplikacije moguće je s alatima za profiliranje, a korištenjem alata Network Monitor generira se detaljan tijek praćenja svih podataka i pristupnih ruta koji prolaze od aplikacije pa do *backend-a* iste. Nama taj alat neće trebati jer naša aplikacija ne koristi nikakvu mrežu. Konfiguracija za izbacivanje (eng. *launch configuration*) kreirana je za svaki projekt te određuje svojstva projekta koja se koristi pri pokretanju i razotkrivanja *bugova* unutar aplikacije. Na primjer imena i lokacije sažetih datoteka SWF

aplikacije nalaze se u konfiguraciji za izbacivanje, a ona se mogu modificirati,tj. mijenjati.

Flash Builder je dostupan u dvije verzije: Standard i Premium. Korištena je Standard verzija koja se razlikuje od Premium u nekoliko stvari, ali ponajviše u tome što nema potpuno automatizirano razvojno okruženje za testiranje memorije, performansi i otkrivanja *bugova*.

5.2. Starling i Feathers framework

Starling je besplatan (eng. *open-source*) okvir otvorenog koda (eng. *framework)* za izradu 2D igara koje se mogu pokrenuti na različitim platformama – bilo da su to različiti preglednici ili mobilne platforme. Starling se bazira u potpunosti na ActionScript 3.0 jeziku te se sav sadržaj renderira direktno kroz grafičku procesnu jedinicu što poboljšava performanse renderiranja.

podržani preglednici	podržane platforme
Opera Mozilla Firefox Safari Internet Explorer Google Chrome	PC Android iOS OS X

Slika 41. podržani preglednici i mobilne platforme

Sadrži cijeli niz alata (eng. *feature*) koji olakšavaju izradu igara – jednostavno kreiranje specijalnih efekata (*blend modes*), animacija (*tweens*), tekstura, raznovrsnih filtera (moguće je kreirati i *custom* filter), kreiranje efekata pomoću čestica, 3D efekata, višestrukih dodirnih točaka (eng. *multitouch*), idr.

Slika 42. http://gamua.com/img/blog/2013/filters.png?mtime=2013-01-14-1216, raznovrsni filteri (blur, sjena, grayscale, mijenjanje boje...)

Kako bi renderiranje bilo brže, Starling kombinira male teksture u veliki atlas tekstura, poznat kao "*sprite sheet*".

Slika 43. kombiniranje više malih tekstura u atlas

Starling je idealan za prikaz aplikacije na različitim rezolucijama jer odabire optimalan set tekstura u odnosu na rezoluciju.Omogućuje da se objekti organiziraju u hijerarhijsku strukturu (relacije *parent-child*) kako bi izrada igara bila intuitivnija.

Brojne igre kreirane su pomoću Starling frameworka, a neke od njih su Angry Birds, Zombie Apocalipse, Kami… Postoji nekoliko biblioteka (eng. *library*) koji proširuju funkcionalnost Starling frameworka, a jedan od njih je i Feathers, koji omogućava kreiranje korisničkih sučelja za mobilne i desktop igre, te aplikacije.

Feathers je besplatna „*open-source*“ biblioteka (eng. *library*) koji sadrži UI (eng. *userinterface*) komponente te podržava interakcije i mišem i dodirom. Zajedno sa Starlingom, služi za razvijanje matičnih (eng. *native*) aplikacija za različite platforme.[10]

5.2.1. Mogućnost više platformi

Kada je kod za određenu aplikaciju napisan unutar Starling okvira, on se zahvaljujući Adobe AIR-u može izvršiti praktički svugdje. Kako je Starling napravljen navrh Adobe Flash tehnologije, on se ne pokreće samo u pregledniku (eng. *browser*) već i na svim većim mobilnim platformama uključujući iOS i Android. To uvelike i pojednostavljuje razvoj video igre, jer nakon otkrivanja grešaka i *bugova*, video igra se instalira na željeni uređaj (računalo, tablet, pametni uređaj, itd.).

5.3. GPU ubrzanje (ubrzanje grafičkom karticom)

Kada se implementira sažeti SWF video u pregledniku, mora se napisati dodatni parametar kako bi se omogućilo Stage3D renderiranje. Stage3D klasa pruža područje gledanja prikaza i programabilan dodatak za renderiranje i iscrtavanje 2D i 3D grafika.

Stage3Dpruža područje visokih performansi za renderiranje sadržaja upotrebljavajući podlogu Context3D klasa. Ta podloga koristi grafičku procesorsku snagu (eng. *graphic processing unit - GPU*) u svakoj situaciji. Dio aplikacije, tj. koda trenutno pokrenut na sceni upotrebljava fiksan broj Stage3D objekata. Broj instanci varira o tipu uređaja na kojemu se pokreće. Stolna (desktop) računala tipično imaju 4 Stage3D instanci.

```
        <!-- Display Resolution for the app (either "standard
        <!-- <requestedDisplayResolution></requestedDisplayRe
    <autoOrients>false</autoOrients>
        <fullScreen>false</fullScreen>
        <visible>true</visible>
        <renderMode>direct</renderMode>
    </initialWindow>
```

Slika 44. mod renderiranja - <renderMode>direct</renderMode>

5.4. XML

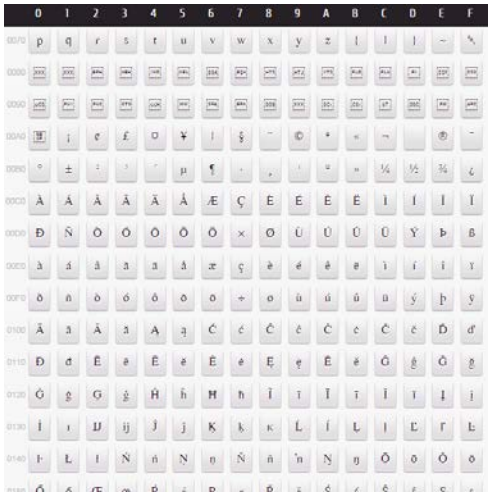
XML je ekstenzija datoteke za *Extensible Markup Language* (XML) format, koji se koristi za stvaranje (pohranu i mogli bi reći "transport") informacija te dijeljenja istih na World Wide Web , intranetu i drugdje, pomoću ASCII teksta.

XML je sličan HTML-u. I XML i HTML sadrže "*mark up*" simbole za opisivanje sadržaja stranice ili datoteke. HTML, međutim, opisuje sadržaj web stranice (uglavnom tekst i grafičke slike) samo u smislu kako se one prikazuju i vrše zajedničku interakciju. Na primjer, slovo p stavljeno u *mark up* oznake pokreće novi paragraf. XML opisuje sadržaj u ovisnosti kakav i koji podatak se opisuje, te na koji način. Na primjer , riječ "*phonenum*" stavljen u *mark up* oznake može ukazivati da su podaci koji slijede broj telefona. XML datoteka može biti obrađena čisto kao podatak u nekom od programa za čitanje XML datoteka ili to može biti pohranjena sa sličnim podacima na drugom računalu ili ipak može biti prikazana kao HTML datoteka. Ovisno o tome kako je neka aplikacija ili program podešen da protumači XML datoteku na računala, taj podatak može biti pohranjen, prikazan ili čak i biran.

Extensible Markup Language je jezik za označavanje koji definira skup pravila za kodiranje dokumenata u formatu koji je čitljiv čovjeku i stroju (pa se može za njega i naći naziv da je to jezik “prijateljski nastrojen” prema stroju, ali i ljudima).

Dizajn ciljevi XML-a naglašavaju jednostavnost, općenitost i upotrebljivost preko interneta. To je ipak tekstualni format podataka uz snažnu potporu putem Unicode-a za različite ljudske jezike. Iako je dizajn XML fokusiran na dokumentima, on se naširoko koristi za zastupanje proizvoljnih struktura podataka, kao što su to oni podaci koji se koriste na web-uslugama.

Po definiciji, XML dokument je niz znakova (Unicode znakovlje). Gotovo svaki legalan Unicode znak se može pojaviti u XML dokumentu.



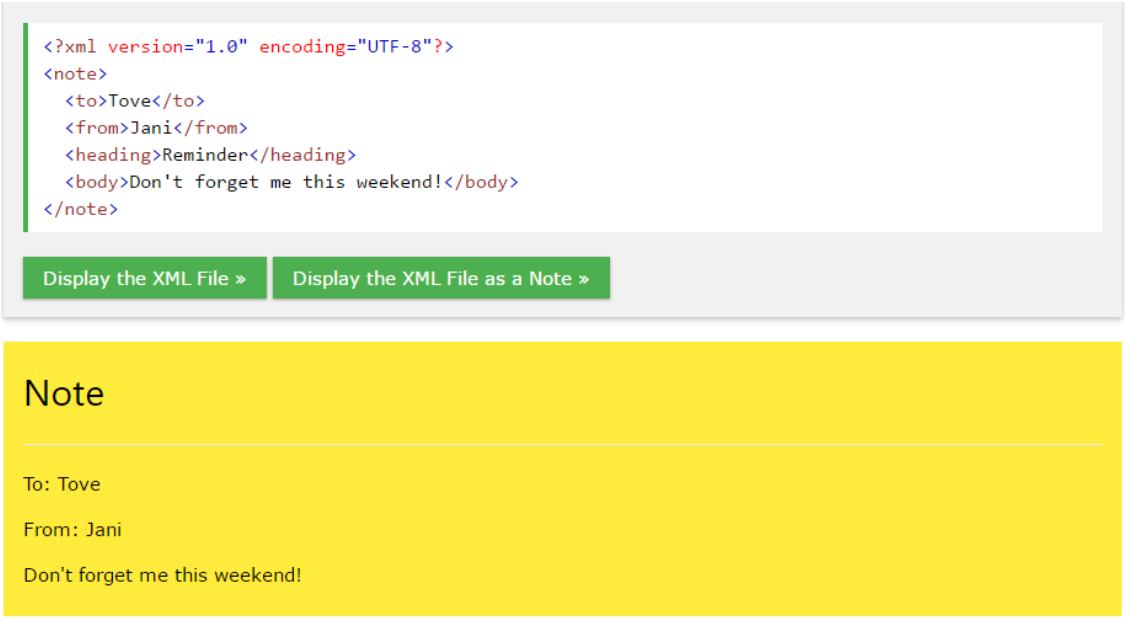
Slika 45. prikaz Unicode znakovlja

Što se tiče procesorske moći i njegove primjene, procesor analizira *mark up* označavanje i prosljeđuje strukturirane podatke aplikaciji. Specifikacija stavlja zahtjeve na ono što XML procesor mora raditi ili ne raditi, ali to se ne primjenjuje na aplikaciju, jer je ona izvan tog djelokruga djelovanja. Procesor (kao specifikacija naziva) se često naziva kolokvijalno kao XML parser.

Znakovi (eng. *characters*) koji čine XML dokument su podijeljeni u sadržajne i one koji označavaju, što se može razlikovati primjenom jednostavnih sintaktičkih pravila. Općenito, dijelovi tj. nizovi koji čine označavanje počinju sa znakom „<“ i završavaju sa istim „>“. Nizovi elemenata koji nisu tako opisani čine sadržaj. Osim toga, razmak prije i poslije krajnjeg elementa klasificira se kao oznaka. Tagovi dolaze u trima svojstvima: Startna oznaka(npr. <*section*>), Krajnja oznaka (npr. *</section>)*,te Prazna oznaka (npr. <*line-break /*>).

Dokumentirani model objekta (*Document Object Model*) je sučeljem orijentirana aplikacija za programiranje sučelja koja omogućuje navigaciju cijelog dokumenta kao da je razgranato drvo gdje te grane predstavljaju sadržaj dokumenta prema nekom slijedu. DOM dokument može biti izrađen od straneanalize sintaksi (eng. *parser*) ili može biti generiran ručno od strane korisnika (uz ograničenja). DOM implementacije imaju tendenciju korištenja velike količine memorije, jer obično zahtijevaju da je cijeli dokument učitan u memoriju i izgrađen kao stablo objekata prije dopuštenog pristup korisniku.

Drugačiji oblik XML procesiranja i obrade API je XML vezivanje podataka, gdje su XML podaci dostupni kao hijerarhija jedinstvenih, snažno upisanih i opisanih klasa, za razliku od generičkih objekata stvorenih od strane analize sintaksi*Document Object Modela*. Ovaj pristup pojednostavljuje razvijanje koda, te u mnogim slučajevima omogućuje identificiranje problema u kodu za vrijeme prevođenja koda (eng. *compiling*), a ne za vrijeme pokrenutog koda (eng. *running time*).



Slika 46. http://www.w3schools.com/xml/default.asp, primjer XML koda

5.4.1. Definiranje početnih parametara pomoću XML-a

Pomoću XML-a definirani su naslovi i podnaslovi igre, opisni dijelovi gumbi i pretežito svi opisni tekstualni dijelovi video igre. Također se pomoći XML parametara mogu brzo definirati i promijeniti jezici koji se koriste u igri. Umjesto pojedinačnog mijenjanja svake riječi, one se promijene u samoj datoteci i automatski prikažu unutar računalne igre. To omogućava veliku brzinu i kohezivnost tekstaunutar igre kao i laku mogućnost višejezičnosti.

```
<?xml version="1.0" encoding="UTF-8"?>
<root>

<!-- Main Menu data -->

<mainScreenTitle>Strigoforia</mainScreenTitle>

<mainScreenDescription>Begin your journey!</mainScreenDescription>


<mainScreenStartButtonLabel>Start</mainScreenStartButtonLabel>

<mainScreenContinueButtonLabel>Continue</mainScreenContinue-
ButtonLabel>

<mainScreenExitButtonLabel>Exit</mainScreenExitButtonLabel>
```

Slika 47. prikaz XML-om definiranih naslova i podnaslova

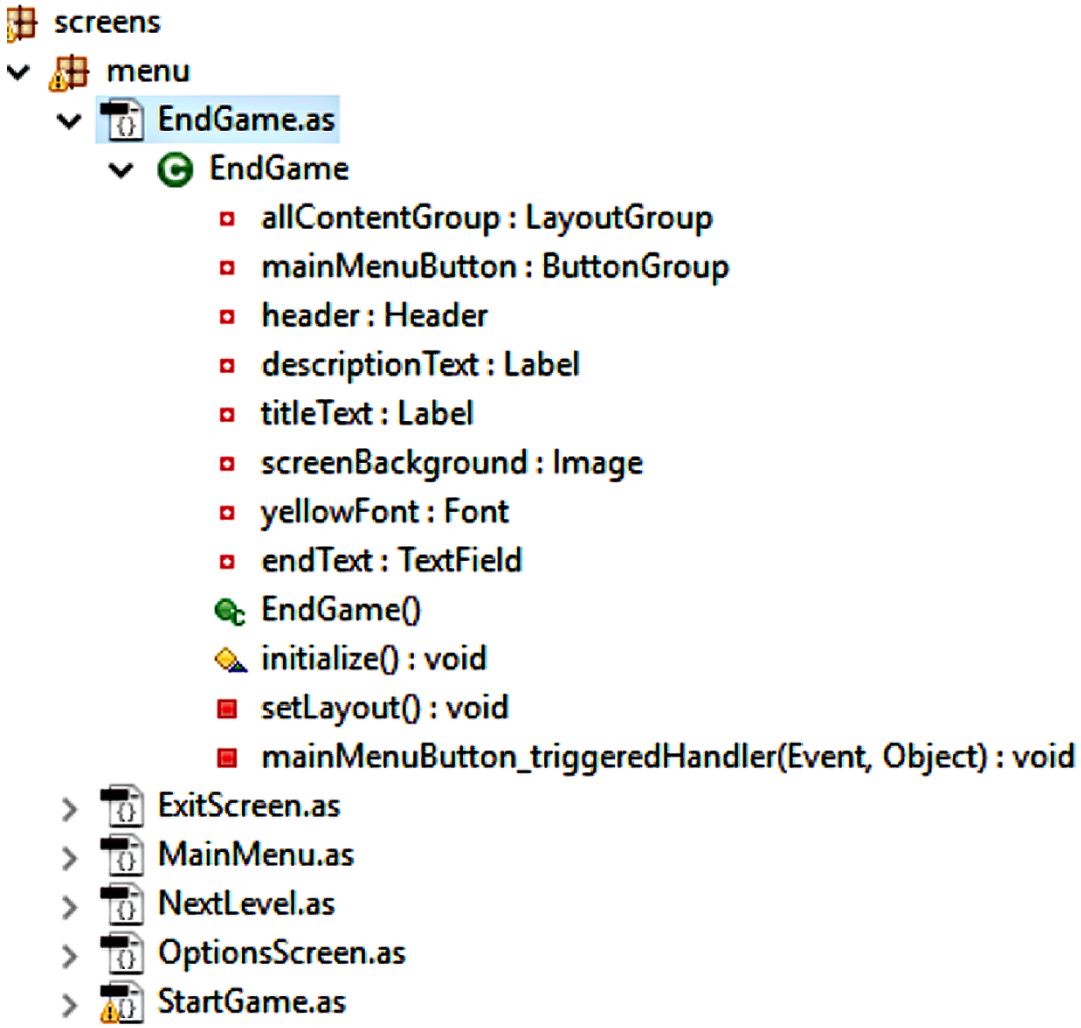
Prikazivanje grafičkih elemenata unutar video igre je olakšano uporabom XML-a, koji u svojoj datoteci ima cijeli opis imena, koordinata i veličine.

```
<?xml version="1.0" encoding="UTF-8"?>
<TextureAtlas imagePath="OWL_Spritesheet.png">
<!--
Created with ShoeBox
http://renderhjs.net/shoebox/
-->

<SubTexture name="Owl-fly_0"  x="814"     y="1540"   width="224"
height="196" frameX="-4" frameY="-11" frameWidth="255" frame-
Height="305"/>
<SubTexture name="Owl-grab_0"       x="0"y="1167"   width="308"
height="127" frameX="-10" frameY="-117" frameWidth="332" frame-
Height="323"/>
<SubTexture name="cricket-walk_4"   x="1411"   y="285"     width="160"
height="85" frameX="-11" frameY="-13" frameWidth="174" frame-
Height="106"/>

</TextureAtlas>
```

Slika 48. prikaz djela koda XML atlas teksture, tzv. „spritesheet“

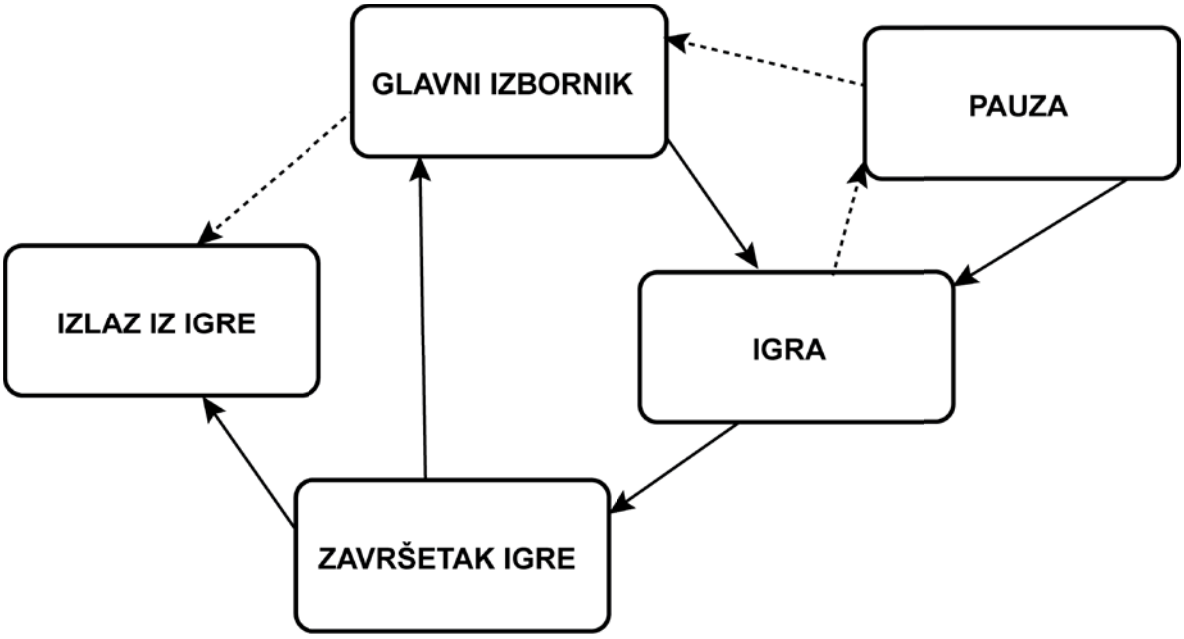
5.4.2. Algoritam beskonačnosti i parametri koji utječu na igru

U video igri stvorena je beskonačna petlja (eng. *infinite loop*) koja baš kao što joj i sam naziv govori neprestano ponavlja istu radnju. Algoritam beskonačnosti ima svrhu da igru balansira po igračevim mogućnostima. Prilikom završetka određene razine pokreće se nova kojoj su ulazni parametri završni rezultat prošle razine. Programski kod koji je zadužen za balansiranje igre je zapravo beskonačna petlja, koja na taj način igru čini zanimljivom i bez kraja. Prednost ovakvog pristupa je to da igrač može igru igrati koliko dugo želi a algoritam se brine da zainteresiranost korisnika ne opada.



Slika 49. prikaz ekrana u mapama Flash Buildera

Kako bi se ta petlja neprestano mogla pokretati unutar video igre napisan je kod i proslijeđen svakom ekranu. Tako je postignut algoritam beskonačnosti. Na slici 50. vidi se prikaz tok ekrana pomoću beskonačne petlje. Broj jedan na slici prikazuje početni ekran, nakon kojeg slijedi sljedeći ekran, ovisno na koju tipku sučelja korisnik pritisne.

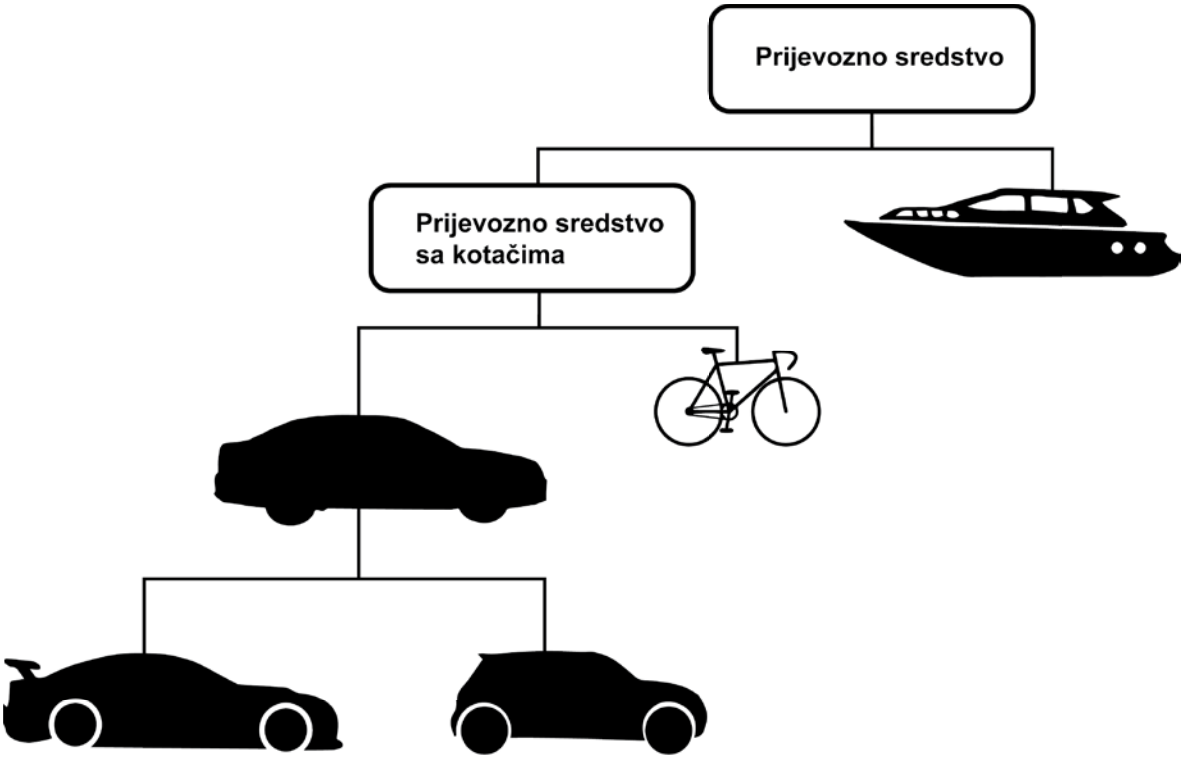


Slika 50. slikovni prikaz algoritma beskonačnosti

5.5. Objektno orijentirano programiranje (OOP - Action Script 3.0)

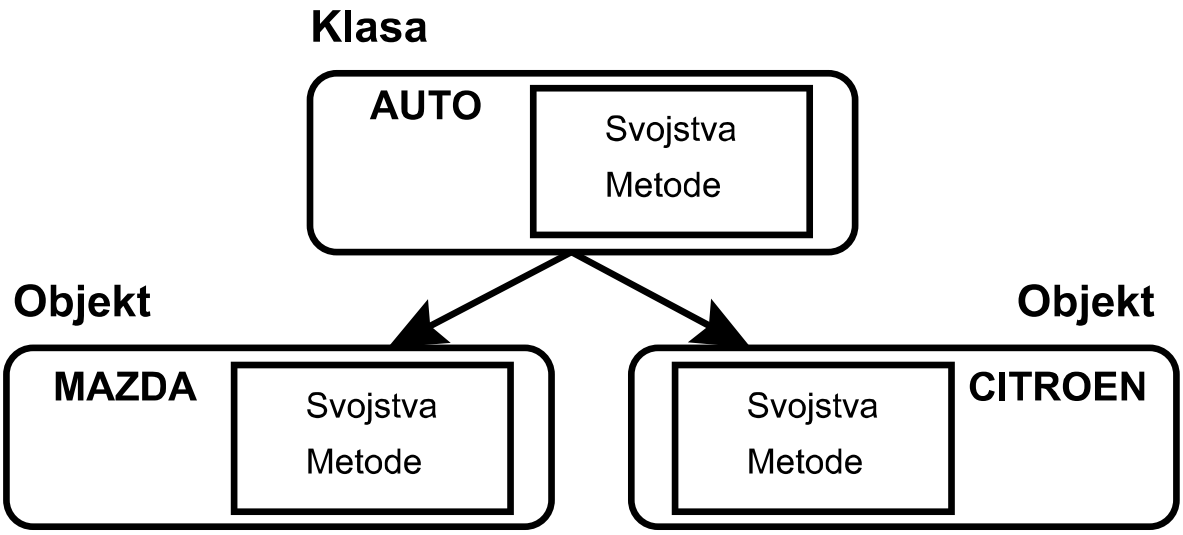
Objektno orijentirano programiranje je postupak izrade programa upotrebom skupa objekata koji međusobno razmjenjuju poruke. Neki od objekata u stvarnom svijetu su: bicikl, motocikl, mobitel, računalo, traktor, sat, isl. Svaki objekt ima najmanje dvije karakteristike koje ga opisuju, a te karakteristike bi bile stanje (eng. *state*) i ponašanje (eng. *behavior*). Na primjeru motocikla mogli bi reći da za opisno stanje možemo uzeti u obzir trenutnu brzinu, trenutnu frekvenciju rotacije oko jedne fiksne osi i broj brzina. Na karakteristikama ponašanje mogli bi dati opisne karakteristike ubrzavanja, usporavanja, kočenja, promjene brzine i skretanja. Dok su objekti u stvarnom okruženju logički opisani, objekti u softverskom okruženju su zapravo entiteti sastavljeni od varijabli (promjenljivih atributa) i pripadnih metoda

modela stvarnih (motocikl, računalo, isl.) ili apstraktnih objekata (događaj, greška, isl.).



Slika 51. primjer OOP programiranja

Varijable (eng. *variables*) su zapravo softverski opis stanja objekta, dok su metode (eng. *methods*) opis za ponašanje objekta (funkcije karakteristične za taj objekt).



Slika 52. prikaz objekta unutar klase

Poruke u sustavu objektno orijentiranog programiranja su mehanizam komunikacije i međusobne interakcije objekata. Prosljeđivanjem tih poruka mogu se obaviti skoro sve vrste interakcija između objekata, a ponašanje unutar mreže objekata definirano je njihovim metodama. Također valja znati da objekti ne moraju biti unutar istog procesa niti na istom računalu da bi slali poruke jedni drugima.

ActionScript 3.0 je objektno orijentiran programski jezik, koji je neovisan o hardveru i softveru. Kompilirani (eng. *compile*) ActionScript kod može se izvršavati na svakoj platformi (hardver + operacijski sustav) na kojoj je instaliran Adobe AIR *runtime*. Razlike između desktop, web i mobilne aplikacije je u samom načinu na koji se izvršava aplikacija, tj. program. Desktop aplikacija je program koji se izvršava na računalu, pod njegovim operacijskim sustavom, za razliku od web i mobilne aplikacije koje se izvršavaju u web pregledniku i na mobilnom uređaju.

5.5.1. MVC obrazac

Većina programskih okvira striktno koristi MVC (eng. *Model View Controller*) tehnologiju. Ta tehnologija je zapravo arhitektonski obrazac korišten u programskoj

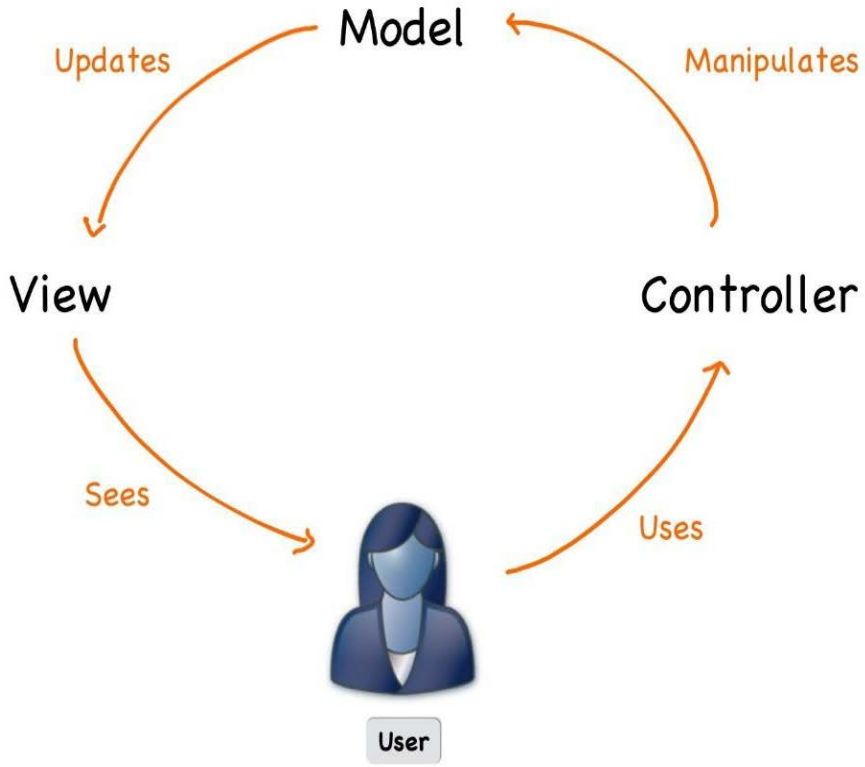
potpori. Koristi se kako bi se razdvojilo korisničko sučelje (eng.*View*) od poslovne logike (eng. *Controller*, upravlja sa korisničkim sučeljem) i modela podataka (eng. *Model*, upravljač baze podataka).[11]

Model je odgovoran za rad s podacima, poput: spajanja i konektiranja na bazu podataka, izvršavanje upita, implementaciju poslovnih pravila, itd. Podaci se prvotno prosljeđuju objektu i upravljaču korisničkog sučelja koji kasnije te podatke prosljeđuje objektu samog korisničkog sučelja. Na taj način, model podataka može „posluživati“ nekoliko objekata koji tada imaju mogućnost prikazivanja tih podataka na različite načine, bez potrebe za redundancijom koda koji je zadužen za rad s podacima. Rezultat takvog pristupa je lakše održavanje koda, smanjenje mogućnosti pojavljivanja grešaka, ponovna upotreba određenog dijela koda (eng*. Code reuse*) itd.

Ukratko, model baze podataka odgovara na zahtjeve koji stižu iz upravljača korisničkog sučelja, priprema ih i obrađuje podatke, te obavještava registrirane objekte samog korisničkog sučelja da ažuriraju svoj prikaz sa novim podacima.

Prednosti korištenja MVC tehnologije su lakše održavanje, testiranje, te nadograđivanje aplikacije. To sve aplikaciju čini proširivom i skalabilnom.

Slika 53. prikazuje grafički princip (način funkcioniranja) MVC tehnologije.



Slika 53. *https://raw.githubusercontent.com/awatson1978/meteor-cookbook/master/images/MVC%20Cycle%20-%20Traditional%20Model.jpg* , MVC tehnologija - struktura

6. REZULTATI I RASPRAVA

6.1. Testiranje i balansiranje parametara računalne igre

Nakon postavljenih parametara unutar videoigre, testiran je odaziv (eng. *responsivness*) same igre na korisnikove upute. Jedan od bitnih parametara je brzina okruženja, tj. pozadine sa parallax efektom, te utjecaj brzine sove i drugih likova u tom okruženju.Sova ima osnovnu, tj. bazičnu brzinu s kojom se prelazi nivo, ali u određenim trenucima dolazi do ubrzanja (pri hvatanju skakavca). U tim trenucima ubrzanje sove se pet puta pomnoži s brojem bazne brzine kako bi se stekao logičan dojam naglog ubrzanja.

Također vrijeme cijele igre je postavljeno na jednu minutu, kako bi korisnik imao osjećaj brzog, tj. žurnog igranja nivoa. Isprobano je više vremenskih ograničenja, od pola pa do pet minuta, ali na kraju je postavljena minuta kao srednja vrijednost.

U konceptualnom djelu začetka ideje o igri, sova je zamišljena da ima 3 života. Tako je ostalo i u završnoj fazi igre. Sve značajne informacije u igri (poput količine života i ostatka vremena) postavljene pregledno kako bi korisniku, tj. igraču bilo lako pratiti stanje igre.

7. ZAKLJUČAK

Sinergijom alata i znanja sa različitih područja kreirana je računalna igra, koja ima dobar temelj i podlogu kako bi korisnik, tj. igrač mogao uživati u njoj. Kroz rad vidimo razne korake koji prikazuju proces nastajanja računalne igre. Ovo je jedan od načina kako se može realizirati video igra, jer svaka vremenska realizacija stvaranja igre ovisi o dogovoru i preferencijama pojedinaca unutar tima. Igra je kreirana sa autorskim slikama i grafičkim elementima, te animirana u posebnom softveru koji je olakšao i skratio dugotrajni proces samog razvoja. Na kraju je uspješno povezana cijela hijerarhija razvoja, od digitalne izrade grafičkih elemenata, animacijskog i programskog dijela te je dobiven konačni rezultat, Strigoforia. Također, vrijedno je naglasiti da iako je ova igra prvenstveno namijenjena kao računalna igra s mogućnošću prilagodbe za druge platforme (Windows, Linux, Mac, isl.), a može se preurediti za mobilnu platformu, poput pametnog telefona (iOS i Android) ili tableta. Tako funkcionalnost igre postaje više platformna. Razvoj ove računalne igre trajao je više mjeseci, ali sa budućim nadogradnjama i razvijanjem igra bi mogla i nadići svoj prvotni potencijal. Trenutno, igra je kreirana i dovršena prema prvotnom zamišljenom planu.

8. LITERATURA

[1] ***https://en.wikipedia.org/wiki/Video_game, 16.07.2016

[2] ***https://en.wikipedia.org/wiki/Super_Mario_Bros, 16.07.2016

[3] ***http://www.webopedia.com/TERM/G/Graphical_User_Interface_GUI.html

[4] Mustač K. (2015), *Izrada multimedijske animacije digitalnih statičnih slika, završni rad,* 16.07.2016

[5] ***http://hr.esotericsoftware.com/spine-in-depth#Što-je-Spine? , 16.07.2016

[6] ***https://www.artrage.com/, 16.07.2016

[7]***http://renderhjs.net/shoebox/, 16.07.2016

[8] ***http://www.adobe.com/products/flash-builder.html, 16.07.2016

[9] Mesić I. (2010), *Programiranje - priručnik, Algebra d.o.o.,* 16.07.2016

[10] ***http://feathersui.com/, 16.07.2016

[11] ***http://docbook.rasip.fer.hr/ddb/res/35/Ch4.html, *Primjena semantičkih formata dokumenata u digitalnim knjižnicama,* 16.07.2016